昭和54年5月2日第3種郵便物認可　平成元年9月1日発行（毎月1回1日発行）第15巻　第9号　通巻148号　ISSN 0387-9569

インターフェース
Interface

9
1989 No.148

特集
カスタマイズ&グレードアップを目指して
続OS-9/68K 徹底解剖（上級編）
●ファイル構造再入門　●ファイル・マネージャの構造と設計
●X68000へのOS-9/68Kの移植　●Internetへの対応/iREX

製作　Turbo Cによるポップアップ・ウィンドウ・ライブラリ
連載　マイコン技術者のための実用的グラフ理論

リで540chまで

ターン・ゼネレータ

るオート・デスキュー機能

荷中

資料請求No.2

インターフェース CONTENTS

特集　続OS-9/68K 徹底解剖（上級編）

一般

連載

〈表紙デザイン〉奥村　恒夫

1989年9月号 No.148

情報ファイル

資料請求No.5

インターフェース 製品別広告索引

製品別広告索引のご案内

インターフェースでは，読者各位に掲載広告を有効活用していただくために，50音別広告索引とともに製品別広告索引を設置しています．これは，小誌だけでなく姉妹誌トランジスタ技術と同一の区分法に基いています．さらに，小社発行半導体情報誌DATUMの目次とも歩調をそろえています．
具体的には，大分類として18項目および中分類として165項目により構成しています．
読者各位が必要とする情報を入手したいときには，製品別広告索引を是非ご活用下さい．
なお，掲載広告についてのお問い合せは添付の資料請求はがきをご利用下さい．(除技術者募集広告)

製品区分表（大分類）

1. 半導体
2. ボード／モジュール
3. 表示部品
4. 受動部品
5. 実装部品
6. 電子材料
7. 電源機器
8. 計測機器
9. 半導体・電子部品製造関連装置
10. その他の電子機器
11. コンピュータ
12. 開発支援機器
13. 周辺機器
14. ソフトウエア
15. その他の材料・部品・機器・装置
16. 流通
17. 情報・教育
18. 案内

資料請求No.7

インターフェース 製品別広告索引

開発支援機器

周辺装置

ソフトウェア

流通

情報，教育

案内

〈主な仕様〉●CPU80286 12MHzノーウェイト●大容量HDD(20MB/40MB)内部増設可能●低消費電力35W●PC-9801Vシリーズ対応の豊富なソフトが活用可能●PC-9800シリーズ対応のオプションボード、周辺機器が使用可能●RAM 640KB標準実装●数値演算プロセッサ80287用ソケット●グラフィック機能●動作中のクロック切換え(6／10／12MHz)●5インチFDD2基標準装備 〈PC-286VF標準価格〉STD：¥298,000／H20：¥423,000／H40：¥513,000

ゆたかな98ソフトを活かす

PC国民機。

※標準価格には、消費税は含まれていません。

●エプソンPCシリーズに関する技術的なご質問・ご相談に電話でお答えします。エプソンPCインフォメーションセンター 東京(03)377-3531 大阪(06)397-0915 ●受付時間/AM9:00～PM5:30 月曜日～金曜日(祝日を除く) ※電話のかけまちがいが増えておりますので、番号をよくお確かめの上おかけ下さい。

エプソン販売株式会社 ●本社:〒151 渋谷区初台1-53-6 ●ショールーム:新宿NSビル5階 ■支店・営業所: ●札幌(011)222-2821 ●仙台(022)263-3691 ●秋田(0188)32-4002 ●酒田(0234)23-8200 ●大宮(048)644-3400 ●千葉(0472)25-0984 ●東京(03)348-6801 ●東京中央(03)258-4841 ●横浜(045)316-4820 ●長野(0262)24-7660 ●松本(0263)36-7251 ●新潟(025)243-8515 ●金沢(0762)62-3216 ●静岡(0542)51-1061 ●名古屋(052)962-7001 ●京都(075)361-7551 ●大阪(06)397-0900 ●大阪南(06)632-3353 ●広島(082)262-5181 ●高松(0878)23-3646 ●福岡(092)471-0761 ●鹿児島(0992)25-7717 ●特販部(03)377-2201

セイコーエプソン株式会社 ●本社:〒392 長野県諏訪市大和3-3-5

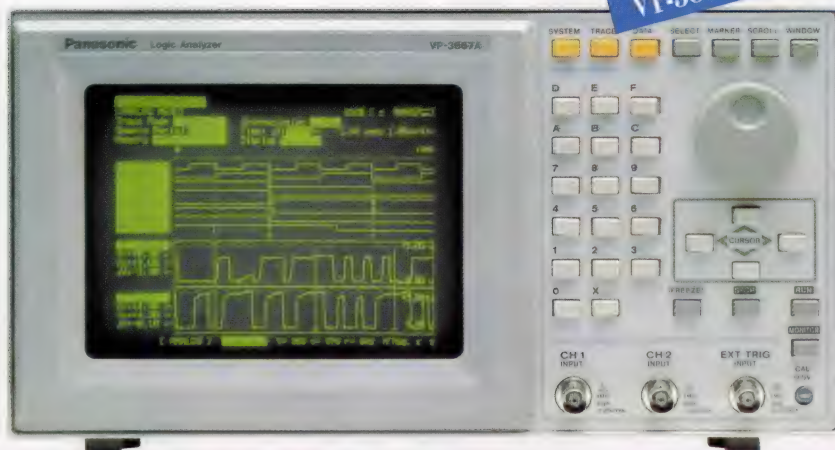
※カタログをご希望の方は、ハガキに資料請求券を貼付の上、住所、氏名、年齢、職業、電話番号および製品名を明記してお申し込みください。送付先:〒151 東京都渋谷区初台1-53-6 エプソン販売株式会社 宣伝部 パソコン資料請求係

PC-286
資料請求券
インターフェイス
9月号

インターフェース 50音別広告索引

Panasonic

パナソニックは、簡単操作です。そして高速200MHz。

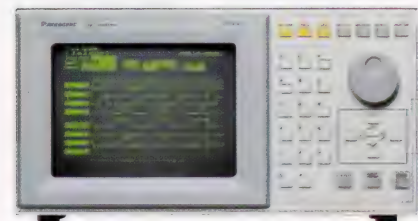
DSO機能搭載
VP-3667A新発売

アナログ解析もできるDSO複合モデル。

充実のロジック機能に加え、DOS機能を搭載。タイミング解析とアナログの実波形観測を同時にこなします。DSO部では、200MSPS・最大2チャンネルでアナログ波形を解析。幅広いロジック観測に応用できます。さらにモニタ機能なら、どんな表示モードのときでもワンタッチでアナログ信号を表示。手軽に波形確認ができます。

標準価格1,850,000円（税別） VP-3667A

ステート解析もできる高機能モデル。

より基本性能を充実。大規模な商品開発や高度な解析に向いています。最大34チャンネル入力、8K bit/ch大容量メモリを実現。また、汎用ステートアナライザとしても使用可能。メモリカード、D•Aモード表示機能も装備。フロントケース内蔵式の小型プリンタ（別売）により、フィールドでも活躍します。

標準価格1,350,000円（税別） VP-3666A

タイミング解析に機能を絞った基本モデル。

タイミング解析機能に徹したタイプ。オシロの延長として、あるいは初めてロジアナを使うかたに適したタイプです。メモリ容量8K bit/ch。最少3nsのグリッチまで検出できます。さらに低価格ながら、スピーディーな障害発見が可能なトリガ機能、水平・垂直拡大機能など、データ解析能力を高める豊富な新機能を搭載。

標準価格850,000円（税別） VP-3665A

※この広告に掲載の全商品のご購入の際には、商品価格の他に別途、消費税額をお支払い下さい

パナソニック ロジックアナライザ

お問い合わせ・資料のご請求は/〒223 横浜市港北区綱島東4丁目3番1号 電話045(531)1231(代表) 松下通信工業株式会社 電子計測事業部

心を満たす先端技術―Human Electronics 松下電器・松下通信

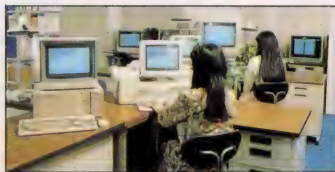
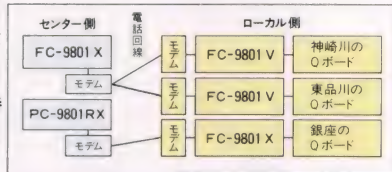
資料請求No. 11

地上高くそびえる大画面、広告ビルボードとして、タイムリーな情報伝達のツールとして注目を浴びるQボード。FAパソコンは、昼夜を問わず刻々と変わる画面を正確にコントロールしています。

情報を広く、タイムリーに伝える新しいメディア、Qボード。

日本アドバンスド・プロダクト㈱のQボードは、世界にも類をみない磁気反転盤による大型ディスプレイボードです。同社が独自に開発したカラープラスティック樹脂製の4色4面体構造をもつ磁気反転素子を組み合わせることにより90色表示が可能です。この磁気反転素子(20㎜～100㎜角)数万個の構成により視認性の高い、鮮やかな画像を多種表示します。Qボードは、国内・国外を問わず数多く設置され、公害などの計測・監視表示に、ニュース・天気予報などの生活情報表示に、さらには屋外の広告表示にと、その利用形態も多岐にわたっています。

画像データの入力、送信、表示コントロールを担うFC-9800シリーズ。

Qボードへの画像表示はまず、ボードの素子構成を考慮した4色の組合せで依頼先からのオーダーに限りなく近い色を再現した下絵を制作します。表示プロセスの第一段階であるこの下絵づくりを担うのが㈱童夢寿(ドムス)のクリエータースタッフです。現在、㈱童夢寿では神崎川(大阪)、東品川、銀座のQボードを担当。下絵づくり、画面表示のオペレーションをPC-9800シリーズとFC-9801Xを使用し行っています。入力された画像データは、それぞれのQボード側に設置されたFC-9800シリーズに電話回線を通じて送られます。データを受け取ったFC-9800シリーズは、秒刻みの緻密なタイムテーブルのもとで数万個に及ぶ磁気反転素子を瞬時に制御しています。

Qボードの下絵の制作、画面表示にたずさわる㈱童夢寿のオペレーションルーム。

屋外設置の多いQボードだけに、耐環境性に優れたFC-9800シリーズが選ばれた。

太陽光の下でこそ高い視認性を発揮し、迫力ある画面を次々と表示するQボードは、おのずとビルの屋上などに設置されることが多くなります。その結果、オペレーションルームから送られてくる画像データを受け取り、表示を行うQボード側のコンピュータにとって、環境はシビアなものとなります。雨、風をしのぐために、建物内に設置されてはいるものの、温度、湿度さらには塵挨に対するきめ細かな配慮が不足しがちな設置環境です。このような悪環境でも24時間連続運転で活躍するのがFC-9800シリーズ。強力ファン、簡易防塵フィルタによる防塵対策、電圧変動や外部ノイズに強い電源の採用など、耐環境性に優れ、信頼性の高いハードウェア仕様が、昼夜を問わず正確な画像、文字情報を表示し続けています。

Qボード側のFC-9801V

さまざまな産業分野でお役にたっています。

FC-9800シリーズ

FC-9801Aの主な特長

●CPUにクロック周波数16MHzの32ビット386を搭載。メモリアクセスノーウェイトにより処理スピードの大幅な高速化を実現。●大容量1.6MバイトのユーザーズメモリMを標準装備。最大12.6Mバイトまで内蔵可能。●CPU本体に1Mバイトタイプの5/3.5インチフロッピィディスク1台のほか、1Mバイトタイプフロッピィディスク相当のRAMファイルを最大2台あるいは20/40Mバイトタイプ3.5インチ固定ディスクが1台内蔵可能。●本体に各種インタフェースボード、PIOボード用の拡張スロットを5スロット内蔵。●信頼性、操作性の高いシステム構築を実現するRAS機能を標準装備。●防塵対策、強力ファン、電圧変動/外部ノイズに強い電源の採用など、耐環境性を強化したハード仕様。●PC-9800シリーズの豊富なハードウェア、ソフトウェア資産を継承。

■FC-9801A設置環境条件

項目＼機種	FC-9801A 本体	FC-9801A FD、HD内蔵時
周囲温度	0～50℃(0～45℃)注1	5～45℃(5～40℃)注1
保存温度 注2	−20～65℃	−20～60℃
湿度(非結露)	20～90%	20～80%
浮遊塵埃	特にひどくないこと(簡易防塵可能)注1	
腐食性ガス	ないこと	
耐震性(16.7HzXYZ各30分)	1.6G	0.5G
耐衝撃性(XYZ方向3回)	10G	10G(FD内蔵時) 2G(HD内蔵時)
電源電圧	AC100V/110V+10%、−15%(AC85～121V)	
電源周波数	50/60Hz±1Hz	
電源雑音	1KVp-p 50ns～1μsパルス	
絶縁抵抗値	20MΩ(DC500V)	
絶縁耐圧	AC1.5KV 1分間	
漏洩電流	1mA以下	
瞬時停電	10ms以下	
接地	第3種	

注1) 簡易防塵フィルタセット装着時
注2) オプションのキーボードの保存温度は−20～50℃

※386は米国インテル社の商標です

理想的なマイコン開発環境"CIDS"

より高度化、複雑化し続けるマイクロコンピュータ開発。限られた期間で最終製品テストまでの開発作業をスムーズに終了する事は至難の技といえます。このため使いやすく機能性、信頼性に優れ、しかも将来の変化に対応すべく拡張性に富んだ開発環境の構築がますます大切となってきました。特にC言語によるコード生成が一般化した現在、実機上でプログラムをソースレベルにて確実に検証できるデバッグ環境と、それを支える信頼のおけるリアルタイムエミュレータが不可欠です。高品質のクロスソフトウエア、エミュレータ、高級言語デバッガを統合化したCIDS開発環境では、デバッグツールの動作確認等に貴重な時間を浪費することなく開発作業そのものへ専念できます。

ES1800リアルタイムエミュレータ

ICEとしてのリアルタイム性、トランスパレンシー性を徹底追求、25MHz68020をはじめ各種の高速16・32ビットCPUを1台でサポートします。

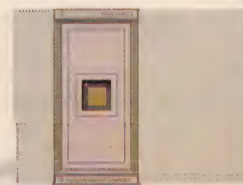
- オブジェクトダウンロード時間を大幅に短縮するSCSIインタフェース(SUNまたはPC-AT互換機用)
- 各種高級言語デバッグをサポート

33MHz68030サポート、EL3200次世代エミュレータ

68030バーストモード、68020を始め、これからの高速CPUの対応も想定した先進のリアルタイムエミュレータです。

▲68030エミュレータ　33MHz
●バーストモードサポート

▲68020エミュレータ　33/25MHz

▲68000エミュレータ　16.7MHz

▲80C186/C188エミュレータ　16MHz
●エンハンストモード完全サポート

高級言語デバッグツール

- VXEL:マイクロテックリサーチ社Mcc68K対応
- X D B:インターメトリックス社InterC対応
- Vscope:OMF86、OMF286サポート

ネットワーク化に最適なCIDS

アプライド・マイクロシステムズのソフトウエアツールは種々のホストコンピュータ、OSをサポート、汎用EWSやPCの豊富なネットワークツールを活用して真の統合化ネットワーク開発環境を構築できます。

主なサポートCPUと保証クロック速度

EL3200	68030	68020	33MHz
ES1800	68020 68010 68000 68008		25MHz 12.5MHz 16.7MHz 10MHz
	80286 80C186 80C86	80C188 80C88	12.5MHz 16.7MHz 10MHz
	Z8000		10MHz

ICEを

Cソースレベル・ビジュアルデバッガ提供。

止めずにどんどん進める。

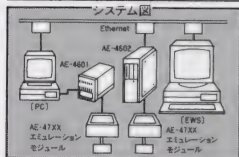
目的とするターゲットがある。必要なICEと開発マシンがある。しかしそれだけで、テーマが達成されるわけではありません。マイコン・ソフト開発は言うまでもなく、決まりきったプロセスが存在します。プログラミング→コンパイル→リンク→デバッグ…。開発を終えるまで、開発者はこれらの作業を何度も繰り返さなくてはならない。そして、クリアしなくてはならない。当然のことながら、開発スピードを高めるために、この繰り返し、つまり"リピート・オペレーション"そのもののサイクルを早めていく操作性が要求されます。しかし、バグを発見する度に、デバッグ外業を中断、わざわざエディタを機動させていたのでは全体としての開発効率は上がらない。バグを発見した時に、いかにデバッグ作業を続行できるか、これがポイントとなります。

そこで私たちはICEのハードウェア性能を100パーセント引き出すHLLDを独自開発し提供することによって、"リピート・オペレーション"のサイクルではなく、リピート回数それ自体の短縮化を実現。これは、デバッグ―つまりHLLD―の高性能化によってのみ可能です。私たちのCソース・レベル・ビジュアル・デバッガ1DEBⅡを用いれば、バグを発見したら、ICEを止めずに瞬時にソース・リストを呼び出す。そしてコンパイル・リンクする。終了後、ワン・タッチ・キーでそのままデバッグを続行。デバッグを中断せずに、どんどん進めてください。デバッグ作業さえ格段に高能率化してしまえば、開発者は自分の開発のイメージ通りに作業を進めることができます。パソコン、そしてEWSによる開発ステージでこのイメージ通りの開発環境の構築を目的としているのが PROICE/Σ(Σ^{32})です。PROICE/Σ(Σ^{32})には、現実に即した開発環境の〈ベスト〉があります。

IWASAKI DEVELOPMENT SYSTEM SUPPORT LIST

CPU	商品名	カバレッジ パフォーマンス	HLLD	アセンブラ	コンパイラ
H16	PROICE/ΣH16	△	ソースレベル デバッガ IDEB ソースレベル ビジュアル デバッガ iDBE-Ⅱ (オプション)	(コンパイラに含む)	日立641016 Cコンパイラ Whitesmith C
H8/500	PROICE/ΣH8/500	△		PROASM-Ⅱ	日立647008 C コンパイラ
H8/300*	PROICE/ΣH8/300	△			×
8086/88	PROICE/Σ86	△		MASM (コンパイラに含む)	Aztec C86 MS-C Lattice C SSI ROMable C Whitesmith C TURBO C PL/M-86
80186/188	PROICE/Σ186	△			
80C186/C188	PROICE/ΣC186	△			
V20/V30	PROICE/ΣV20	△			
V25/V35	PROICE/ΣV25	△			
V40/V50	PROICE/ΣV40	△			
68000/10	PROICE/Σ68K	△		(コンパイラに含む)	MCC68K
68020	PROICE/Σ32 68020	△			Whitesmith C

△:オプション　×:不可　*開発中

高速ネットワークに対応します。〝システムエミュレータAE-4600シリーズ〟。

システムエミュレータAE-4600シリーズは、パソコンをホストとする〈AE-4601〉とイーサネットに直結できる〈AE-4602〉の2機種。それぞれハイレベルランゲージによる言語処理からデバッグまでをトータルにサポートします。■**高級言語**：高級言語とアセンブラの両方に対応したデバッグ環境の要求によりインターメトリックス社のXDBやマイクロテックリサーチ社のXRAYなどのソフトウェアツールをサポートしています。これらクロスソフトやHLLDにより高い自由度をもった開発環境を提供します。■**シンボリックデバッグ**：プログラムのダウンロードをしながらすべてのシンボル付命令ステップにブレークポイントを自動設定します。なお、指定アドレス間にブレーク設定可能なレンジブレーク、4レベルまで設定可能なシーケンシャルブレーク、またリアルタイム・パスカウントも実現しています。■**カバレッジ・パフォーマンス**：カバレッジはC0カバレッジをサポートし、特に「実行した部分のみをリポート」「実行していない部分のみをリポート」「64Kバイト・アドレス空間のテスト」の3点に絞っています。これらをカバレッジコマンドによりアドレス、イベントの双方で設定できます。パフォーマンス機能では「2点間の実行時間や実行回数を測定する」ことによりプログラム実行時のモジュールのCPU占有率をチェックします。パフォーマンスコマンドはアドレス、イベントの双方で設定できます。［AE-4601］●PC-98シリーズなどのお手持ちのパーソナルコンピュータが、ホストとして利用できます。●AE-465X共通モジュールを交換することによって、8bit、16bit、32bitまで簡単に拡張することができます。エミュレーションモジュールは、AE-47XXを使用します。(AE-4601、AE-4602共通)●ホストとの接続は専用パラレルI/Fによって高速通信が可能です。またホストにイーサネットボードを装着することによってネットワーク対応も容易に実現できます。［AE-4602］●ネットワークに直結でき、高速処理を実現します。●ネットワーク上の複数のワークステーション、パーソナルコンピュータから1台のAE-4602を操作することができます。●ネットワーク上の複数のAE-4602を1台のワークステーション、パーソナルコンピュータから操作することができます。●ホストとAE-47XXのエミュレーションメモリ、ターゲットメモリ間のデータ転送が可能です。●強力なトリガとともに、カバレッジ/パフォーマンス機能を標準でサポートします。

高性能を徹底して追求しました。〝デバッグステーションAE-4133〟。

デバッグステーションAE-4133は、機動性を重視したオールインワンタイプで、68020をメインCPU にOS-9上でUNIXライクな高機能・高性能デバッグを実現します。■**操作性に優れたマンマシンインタフェース**：ファンクションメニューキー、フルスクリーンモード、フリースタイルコマンド、ヒストリ機能、ロギングなど使いやすい機能を提供します。■**高級言語デバッグ**：Cコンパイラで開発したプログラムのソース行単位の表示、実行、停止、ステップ動作やプロンジャのトレースが可能です。■**リアルタイムトレース**：●32ビットCPUに対応する76ビット×16Kワードの大容量。●クォリファイトリガを使用した条件トレースが可能。●プログラム実行中にトレース内容を表示。●表示内容を選択できるセレクト機能。■**トリガ**：CPUのストップ、トレース制御、タイマ制御、外部信号出力など4機能8種類をサポート。トリガ発生条件として汎用イベント(8チャンネル)やシーケンスイベント(1チャンネル16レベル)などがあります。■**実行ブレーク**：高機能CPU特有のプリフェッチ機能に束縛されないブレーク設定ができます。●ブレークポイントとしてポイントをフルアドレスで設定できます。■**シンボリックデバッグ**：シンボルとして高級言語を意識しており、ソースライン、ファンクション名、アーギュメント、グローバル/ローカルスタテック変数、自動変数、構造体などが使用可能です。また複数のロードモジュールのシンボルが同時に使用できます。

☆8～32ビットまで、サポートMPUが豊富です。Z80、8085、6801、6301、64180、8086/88、68000/10、80186/188、V40、V50、V33、68020、80386、68030。開発中ならびに本体により接続の異なるものがあります。詳細につきましては別途お問い合わせください。

☆**主な仕様** ［AE-4133］CPU：68020/メモリ：4Mバイト/HDD：20Mバイト標準/FDD：5"2HD1台/10インチCRT内蔵/RS-232C×2/セントロプリンタ/VT100ターミナルモード標準 ［AE-4601］ホスト：PC-98シリーズ、LUNA、IBM PC-AT、IF-800、RX-120、FM-Rシリーズ、J-3100シリーズ(MS-DOS V3.1以上)/インタフェース：専用パラレルI/F ［AE-4602］ホスト：PC-98シリーズ、IBM PC-AT、SUN-3、NEWS(UNIXまたはMS-DOS V3.1以上)/インタフェース：Ethernet(TCP-IP)

(022)262-2637・茨城(0292)27-1271・北関東(048)645-7744・西東京(0425)28-1221
神戸(078)371-1301・広島(082)245-2663・福岡(092)472-2318・熊本(096)357-1930

※開発ツールのお問い合せは、MCグループまで。：フリーダイヤル0120(40)0211

digital
日本DEC

※CPU本体、15インチ・モノクロモニタ、
キーボード、マウス、UNIXライセンスを
含む基本構成の価格です。

10MIPS

98.8万円

DECstation 3100の低価格版。
業界初のプライスパフォーマンスを実現。

新発売

エントリーレベルのRISC/UNIXワークステーション

DECstation 2100

CPU、FPUにはDECstation 3100(14.3MIPS)と同じMIPS*社の32ビット超高速RISCチップR2000、R2010を採用。10.4MIPS、Linpackベンチマークで2.7MFLOPS(単精度浮動小数点演算)、1.2MFLOPS(倍精度浮動小数点演算)の高速処理を実現。

仕様 クロック/12.5MHz メモリ容量/8〜24MB ディスク容量/104〜1,204MB
グラフィック・モニタ/15と19インチ モノクロとカラー 解像度/1,024×864ピクセル
100万インストラクション/秒(Dhrystone、GREP、YACC、DIFFとNROFFのベンチマークの組み合わせの数値)

＊MIPSは米国MIPS Computer Systems社の商標です。＊UNIXオペレーティング・システムは米国AT&Tが開発しライセンスしています。＊NFSは米国Sun Microsystems社の商標です

日本 ディジタル イクイップメント株式会社

本社／〒170 東京都豊島区東池袋3-1-1サンシャイン60 ☎03(989)7111(代表)

お問い合わせ・資料請求は本社広報宣伝部へ

大阪支店 ☎06(222)9000 営業所：札幌☎011(271)6681／仙台☎022(265)5476 筑波☎0298(52)3311 大宮☎048(645)2655 日比谷☎03(595)2011 千代田☎03(259)5211／飯田橋☎03(5261)3111／八王子☎0426(42)8231／横浜☎045(441)8666 厚木☎0462(29)3771／浜松☎0534(56)1277／名古屋☎052(586)2811 豊田☎0565(35)3231／松本☎0263(33)6777 京都☎075(211)4811 広島☎082(223)7127 福岡☎092(482)2811／長崎☎0958(27)3832 熊本☎096(353)2944

アナログ技術の精鋭

MAXIM

マキシム社(U.S.A)

積分型A/D MAX133/134

MAXIM MAX134

MAX133/134はマイクロプロセッサ・インターフェースを内蔵した40,000カウント分解能積分型A/Dコンバータです。

MAX133/134はDC/AC 電圧、電流はもちろん抵抗も測定でき、またマイクロプロセッサコントロールにより、その入力レンジを切換えたり、警告音を出したりすることも可能です。
〔例〕±400,0mV～±4000V〕
すなわち、これ一つであなたのコンピュータシステムも、40,000カウントディジタルマルチメータに変身することができます。
マルチプレクサを必要としない場合は、このマルチプレクサを一般の信号切換用マルチプレクサとしても利用できます。

※MAX134にはIBM・PC/AT対応の評価用キットが用意されています。

主な特徴

- ●40,000カウント分解能
- ●5μVまで分解
- ●0.05%精度
- ●変換時間　20回／秒
- ●マイクロプロセッサ・インターフェース
- ●消費電流　100μA

販売代理店
インターニックス株式会社

●資料が用意されていますのでご請求ください。なお、技術的なお問合わせは、半導体マーケティング1部DA課まで(☎03-369-1105)お待ちしています。

●本社 〒160 東京都新宿区西新宿7-4-7 新宿浜田ビル ☎03(369)1101
●関西☎06(364)5971 ●厚木☎0462(21)1334 ●八王子☎0426(45)8371 ●長野☎0268(25)1610 ●名古屋☎052(452)8841 ●福岡☎092(472)7716

資料請求No.24

VMEボードといえば

フォースコンピュータ社

専用ゲートアレイ搭載

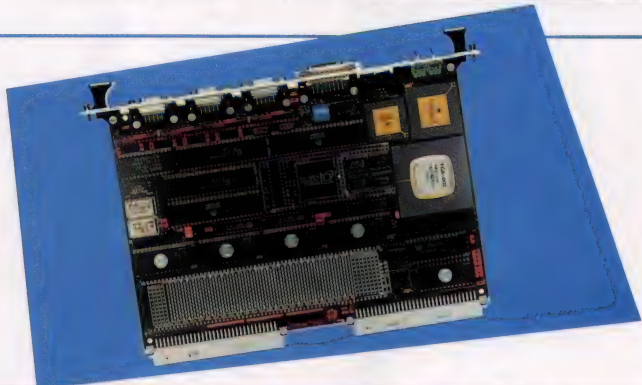
(32bit DMAコントローラ、メッセージ・ブロードキャスト)

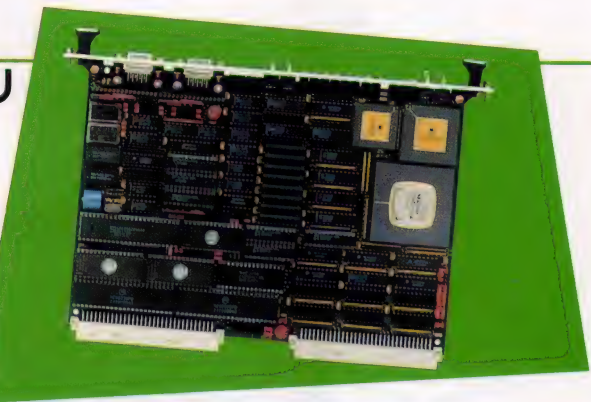
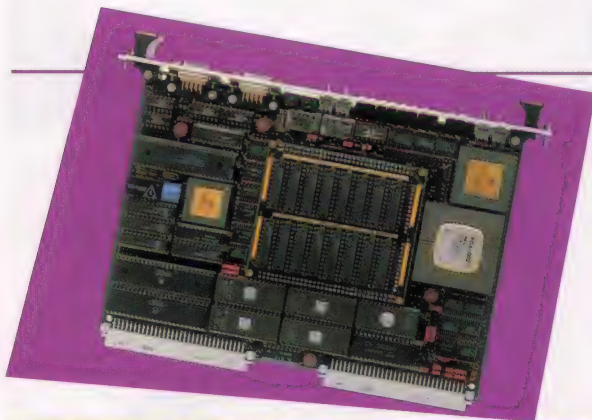
VME bus 68030 CPUモジュール3機種

新登場

- ■32bit DMAコントローラ
- ■2 メッセージ・ブロードキャスト
- ■8 メイルボックス・インタラプト
- ■システム・ブートEPROM
- ■32KByte SRAM
  バッテリ・バックアップ
- ■リアルタイム・クロック
  バッテリ・バックアップ
- ■32bit VMEbus
  マスタ・スレーブ インタフェイス
- ■VMEPROM
  リアルタイム・モニタ標準搭載
  ライセンス不要
  和文マニュアル付き

CPU-30 ザ・サラブレッド

- ■68030:20MHz, 25MHz
- ■68882:20MHz, 25MHz
- ■4MByte デュアルポートDRAM
  バーストフィルモード・サポート
- ■4 EPROMソケット(最大4MByte)
- ■4 シリアルI/O
- ■2 24bitタイマ
  1 8bitタイマ
- ■SCSI bus インタフェイス
- ■FDDインタフェイス SA460準拠
- ■Ethernet モジュール
  64KByte専用バッファ付き
- ■8bit パラレルI/O

CPU-31 ザ・シミュレータ

- ■68030:16.7MHz, 20MHz
- ■68882:16.7MHz, 20MHz
- ■256KByte
  1MByte デュアルポートSRAM
  真のノーウェイト動作
  (シンクロナス・デュアルポート
  SRAM)
- ■4 EPROMソケット(4最大4MByte)
- ■2 シリアルI/O
- ■2 24Bitタイマ
  4 8bitタイマ
- ■VSBインタフェイス

CPU-33 ザ・ローコストエントリ

- ■68030:16.7MHz, 25MHz
- ■68882:16.7MHz, 25MHz
- ■1MByte デュアルポートDRAM
- ■2 EPROMソケット(最大2MByte)
- ■2 シリアルI/O
- ■8bit パラレルI/O
- ■2 24bitタイマ
  1 8bitタイマ

■低価格

¥318,000より

日本総代理店

インターニックス株式会社

●資料が用意されていますのでご請求ください。なお、技術的なお問合わせは、営業部システム課まで(☎03-369-1104)お待ちしております。

●本社 〒160 東京都新宿区西新宿7-4-7 新宿浜田ビル ☎03(369)1101
●関西☎06(364)5971 ●厚木☎0462(21)1334 ●八王子☎0426(45)8371 ●長野☎0268(25)1610 ●名古屋☎052(452)8841 ●福岡☎092(472)7716

資料請求No.25

システム構築の自由度を広げるTVMEシリーズ
VME攻略
TVME
SERIES
一枚のVMEボード上に高いパフォーマンスを実現させてきたTVMEシリーズ。Gmicro CPU:H32/200を搭載したTRON仕様モジュールをはじめメモリモジュール、画像入力・モジュール、DSPモジュール、PIA・モジュール、インテリジェント・通信・モジュールなど機種も充実。オペレーティング・システムOS-9に加えて、OS-9用の各種オリジナルドライバ、アプリケーションソフト等も用意。アバールデータのTVMEシリーズは、用途に合わせた自由なシステム構築を可能にします。
I/O Channel Bus
VME Bus
VME-32 Bus
Local Bus
VSB
68000 MPU モジュール
TVME-100c
●68000(10MHz)●ローカルメモリ:28PinソケットBケ●RS-232C×2●I/O channel Bus サポート
※アドオンモジュール(TSMシリーズ)拡張可
メインコンソール
FDC モジュール
(I/Oチャネル)
TVME-400
●3.5/5.25、8インチFDD同タイプを4台まで接続●FDDコントローラ:μPD765AC●VFO回路内蔵
PIA モジュール
(I/Oチャネル)
TVME-410
●PIA:68B21×2●入出力:8Bit×4ポート(TTLレベル)●割込入力:各ポート2ライン
68000 MPU モジュール
TVME-110
●68000(12.5MHz)●ローカルメモリ:28PinRAMソケット2ケ●デュアルポートメモリ:1MByte●RS-232C×2●プリンタI/F×1
メインコンソール
プリンタ
I/Oチャネルコンバータ モジュール
TVME-390
●プログラマブルに割り込みレベル設定可能●バッテリバックアップ回路内蔵●RTC内蔵
インテリジェント シリアルインターフェース モジュール
TVME-350
●ローカルCPU68008●RS-232C×4CH(RS-422×2)●各チャンネル入出力バッファ付(計60KByte)●パラレル1ch
パラレルI/F
OS-9
システム モジュール
TVME-170
(OS-9/70 PACK)
●ローカルメモリ:40PinROMソケット4ケ、28/32PinRAMソケット2ケ●DRAM:1Mor2MByte●RS-232C×2●プリンタI/F×1●SCSI×1●3.5"FDD I/F×1
※OS-9をセットにしたお買得版
FDD
HDD
CCDカメラインターフェース モジュール
TVME-330
●CCDカメラ2台まで接続●フレームメモリ:512×512×2bit●画像入力:2値化方式●CRT出力:コンポジット出力
CCDカメラ
フォトアイソレーション モジュール
TVME-340
●入力部:定格入力電流10mA、割込入力4bit●出力部:定格出力電圧DC12V～24V、定格出力電流0.2A/点、オープンコレクタ16点×2ポート●LEDインジケータ*TSM-340
TTLパラレルI/O モジュール
TVME-342
●PIA:68B21×4●入出力:8bit×8ポート(TTLレベル)●割込入力:各ポート2ライン
入出力
68020 MPU モジュール
TVME-124
●68020(16.67MHz)●コプロセッサ:68881*●ローカルメモリ:40PinROMソケット6ケ、28/32PinRAMソケット2ケ●デュアルポートメモリ:1M*、2M*、4M* or8M*Byte(ノーウエイト)●RS-232C×2●プリンタI/F×1
TSM-124
1MByte
D SRAMメモ
FDD/SCSI
コントローラ モジュール
TVME-322
●3.5、5.25、8インチFDD同タイプを4台まで接続●SCSIインターフェース●VFO回路内蔵●VME32 Bit bus
SCSI
画像入力 モジュール
TVME-336
●カメラ2台まで接続●カラービデオ出力回路内蔵●フレームメモリ:512×512×12bit×2面●入力分解能8bit●DSPローカルバス(TVME-337用I/F)
DSP モジュール
TVME-337
●DSP:TMS320C25×2●データメモリ:32KWORD×2●プログラムメモリ:ROM/RAM28Pinソケット×4●グローバルメモリ:SRAM32Pinソケット×16●DSPローカルバス(TVME-336用I/F)
68020 MPU
モジュール
TVME-122
●68020(16.67MHz)●コプロセッサ:68881*●デュアルポートメモリ:1Mor4MByte*●RS-232C×2●ローカルメモリ:ROM128KByte、RAM64KByte●プリンタI/F×1
CRT
キーボード
セントロI/F
マウス
テレビモニタ
グラフィックコンソール モジュール
TVME-300
●表示能力:グラフィック640×200ドット、文字80字×25行●ビデオRAM128KByte●プリンタI/F×1
グラフィックコンソール モジュール
TVME-310
●解像度1024×750ドット●4096色中16色同時表示●スクリーンメモリ384Kバイト●ローカルメモリ512K・2MByte●キーボードI/F●マウスI/F●漢字ROM*
イメージプロセッシング モジュール
TVME-334
●入力部:NTSCモノクロ・カラー●フレームメモリ:RGB(512×512×6)、オーバーレイ(512×512×3)、ピクセルプロテクト(512×512×1)●DSP:TMS320C25●出力部:RGB各6Bit・SYNC信号、出力ルックアップテーブル512×6Bit
68020・68030 MPU
モジュール
TVME-120
●68020or68030*(16.67MHz)●コプロセッサ:68881*●デュアルポートメモリ:1Mor4MByte●プリンタI/F×1●RS-232C×2(RS-422対応)●VSBサポート●ローカルメモリ:ROM128KB、RAM64KB
4M/8MBメモリ モジュール
TVME-220
●容量8MByte(4MByteタイプ有)●パリティチェック機能●メモリ・インヒビット機能●VME高速アクセス・モード機能●VME/VSB32 bit bus
2MBメモリ モジュール
TVME-222
●容量2MByte●パリティチェック機能●メモリ・インヒビット機能●VME高速アクセス・モード機能
FDD/HDDコントローラ
モジュール
TVME-320
●3.5/5.25、8インチFDD同タイプを4台まで接続●ANSI規格FDDサポート●SASI I/F●VFO回路内蔵
フロッピーディスク
SASI
ハードディスク
コントローラ
NDC3011
ハードディスク
Gmicro MPU
モジュール
TVME-150
●MPU:GmicroH32/200(20MHz)●RS-232C×2●ローカルメモリ:ROM2MByte(最大)、RAM128KByte●DRAM:1M、2M、4M or 8MByte●プリンタI/F×1
※TRON仕様OS移植中
プリンタ
メインコンソール
*印はオプション

強力なPLDコンパイラと先進プログラマのコンビネーション

PLD戦略

PLDの属性を最大限に活用可能にする強力な論理圧縮機能を持ち、最適化を高速で行うPLDコンパイラLOG/iC。幅広い書き込み対象を持ち、プログラマとして高い信頼性と多彩な機能を持つPECKER-30。この強力なコンパイラとプログラマの組合せは、論理設計からPLDへのプログラムまでをパーソナルユースで実現し、PLD開発をさらにやさしくします。

新世代PLDコンパイラ登場

LOG/iC

ロジック

LOG/iCの特長

●ブール代数式・真理値表・状態遷移図など各種シンタックスをサポート●強力な論理圧縮●最適化がスピーディ●PLDの種類に捕らわれることなく設計が自由●最適なデバイスはLOG/iCが自動的に選択●ファンクションベリファイアでプログラム前に動作チェックOK●論理設計と物理的設計を分離したので新デバイスに対しても今までのデータが利用可能●テストベクタ自動生成●動作環境/NEC社PC9081、IBM-PC/AT/XT、HP9000、VAX、APOLLO その他

THE SYSTEM PROGRAMMER PECKER30

ADAPTER-Aをセットすれば、PECKER-30は各デバイスメーカーのPLD・EPLDに対応。PLDコンパイラ「LOG/iC」により生成されたヒューズパターンとテストベクタをJEDECフォーマットで受け、デバイスにプログラミングします。もちろんターミナルを使ってのヒューズプロットの編集も可能です。

ADAPTER-A書き込み対象メーカー：ALTERA、AMD/MMI、ATMEL、CYPRESS、EXEL、FUJITSU、HYUNDAI、ICT、INTEL、LATTICE、NS、RICOH、SIGNETICS、TI、VTI

※各社から販売されているPLDコンパイラもサポートしています。

PECKER-30 の特長

●バッファRAM標準2Mbit搭載(8Mbit搭載タイプ有)、最大32Mbitまで拡張可能●システムROM交換可能●各社高速アルゴリズム対応●60種以上のコマンド群●フルリモートコントロールできるオンライン機能●各種チェック機能●80文字表示LCD採用●簡単なアダプタ交換●RS-232C、セントロニクスI/F各1ch標準装備●各種転送フォーマット対応

PECKER-30のアダプタ

ADAPTER-B：〔MOS対応〕・EPROM/16K～4Mbit・EEPROM/16K～256Kbit・1ChipCPU/8742、8742AH、8748、8748H、8741、8741A、8741AH、8749H、8755A、8751、8751H、87C51、8752BH・ハイスピードPROM/WSI、ICT、SEEQ、MOTOROLA、TI、ATMEL、CYPRESS、GOLDSTAR、EXEL、SHARP、RICOH

ADAPTER-C：〔バイポーラPROM〕AMD(MMI)、FUJITSU、MITUBISHI、NEC、SIGNETICS、NS、SGS-THOMSON、TI

ADAPTER-D：〔マルチセット対応〕・同時8個書き・EPROM/16K～1Mbit

ADAPTER-E：〔ROMエミュレータ〕・エミューレーションRAM/最大256KByte

FLASH PROGRAMMER PECKER11

プログラミングに関わるすべての機能を高速化したフラッシュプログラマ。高速性を要求されるメガビット時代に応えます。書き込みデバイスも幅広くサポート。80文字表示LCD採用やヘルプ表示など操作性もアップ。アダプタやシステムの交換を簡単にし拡張性を高めた新プログラマです。

PECKER-11の特長

●バッファRAM標準2Mbit搭載（8Mbit搭載タイプ有）●システムROM交換可能●各社高速アルゴリズム対応●フルリモートコントロールのオワライン機能●各種チェック機能●80文字表示LCD●アダプタ交換簡単●RS-232C、セントロニクスI/F各1ch標準装備●40種を超えるコマンド●各種転送フォーマット対応●高速データ転送対応（PC9801よりペッカーへのデータ転送時間1Mbitで約30秒）

PECKER-11のアダプタ

RX1：〔EPROM対応〕・EPROM/16K～2Mbit、EEPROM/16K～256Kbit

RX2：〔1chipCPU対応〕8741、8741H、8741AH、8742、8742AH、8748、8749H、8755、8751、8751H、87C51、8752BH

RX3：〔ハイスピードEPROM対応〕WSI、ICT、SEEQ、MOTOROLA、TI、ATMEL、CYPRESS、GOLDSTAR、EXEL、SHARP、RICOH

RX30：〔EPLD対応〕ALTERA、FUJITSU、ICT、INTEL、LATTICE、RICOH、VTI、ATMEL、CYPRESS、EXEL、AMD/MMI

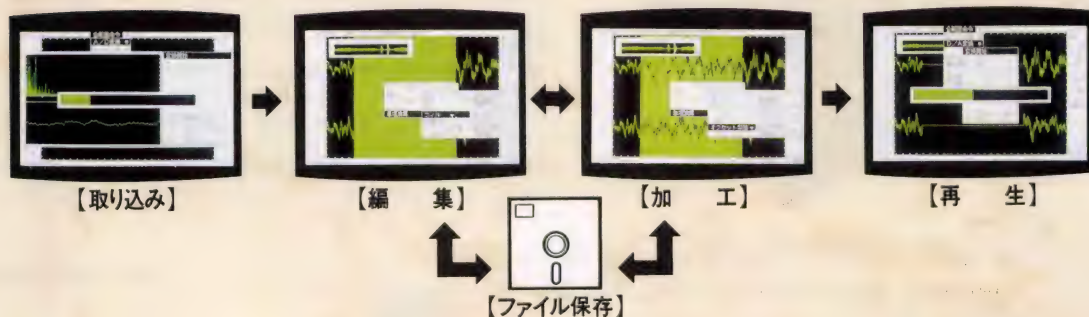
RX40：〔マルチセット対応〕・同時4個書き・EPROM/16K～2Mbit、EEPROM/16K～256Kbit

LOG/iCはISDATA社の登録商標です。

東京支店／〒160 東京都新宿区西新宿6-16-6 第3丸善ビル
TEL.(03)344-2001 FAX.(03)344-2007

このスペックで、この価格。

プロは言うに及ばず、アマチュアレベルまで、広く「音を操る」ことにかけては高い評価が寄せられている"音専科Sound Master"。ワンボードに16ビット高速A/D・D/A＋フィルタを搭載。さらに使いやすいソフトウェアまで付属してこの価格。データ収集からFFT分析、ディジタル編集・加工・保存・再生まですべてPC-9801上でしかもプロスペックでこなすSound Masterは、音に係わるすべてのユーザ必携のシステムなのです。

音を操る人のシステムです。

Sound Master

2ch/DMA16bit A/D＋D/A＋Filter＋Soft ware

¥98,000

●型番 CAB-1123-01-13××

音声・音響などのアナログデータ収集から FFT分析、ディジタル編集・加工・保存・再生まで PC-9801上で実現します。

❶SOUND MASTER(サウンドマスター)は新開発16ビット2チャネルA/D＋D/A＋フィルタ・ボードとコントロール・ソフトのセットシステムです。

❷SOUND MASTERは音声・音響などのアナログデータの収集からFFT分析・ディジタル編集・加工・保存・再生まで、すべてをPC-9801上で実現します。

❸SOUND MASTERは音声・音響の分析・合成システムとして入門から実践まで、すべてのステージを完璧にサポートしています。

❹SOUND MASTERでディジタル・サウンドの新時代を体感してください。

■ソフトウェア

●2チャネルの信号を32Mバイトまで連続して入出力できます。●収集前に波形をリアルタイムにモニターできます。●収集したデータを、自由な形式で高速描画できます。●収集したデータの波形の編集・加工など高機能波形エディットができます。●周波数スペクトラムをマルチウィンドウ形式で表示できます。●収集したデータをファイルに保存することができます。●マニュアルは親切なユーザーズマニュアルとハードウェアマニュアルがついています。ユーザーズマニュアルの巻末には、入門者向きに用語集を載せています。

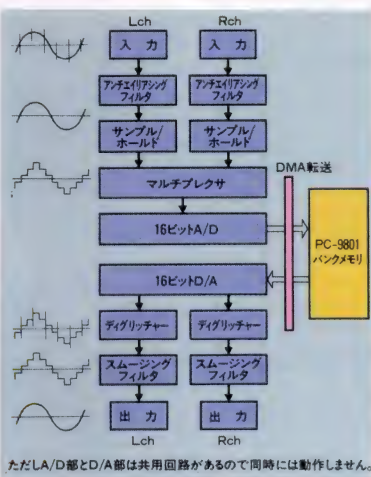
■ハードウェア

●A/D、D/Aは16ビットの高分解能。(DATと同じ)

●2チャネル同時サンプリング。アンチエイリアシング・フィルタ、スムージング・フィルタ用に20kHz 7次ローパスフィルタ内蔵。●サンプリング周波数はDATと同じ32kHz、44.1kHz、48kHz及び外部クロックをソフト選択できます。(1チャネル時は2倍オーバー・サンプリング可能)●周波数帯域/5Hz～20kHz●DMA転送可能●ステレオアンプとピンプラグコードで接続するだけ。

■ハードウェアのみで使用可能

●サンプルソフト付属●DMA転送はバイト・ショート・ワード転送をサポートしており、ワード転送で200kHz/チャネル可能(機種はVM21、VX、UX、XA、XL、XL²、RL、RA、RX、ES、EX)●全てのPC-9801で100kHz/チャネル可能

ただしA/D部とD/A部は共用回路があるので同時には動作しません。

動作環境

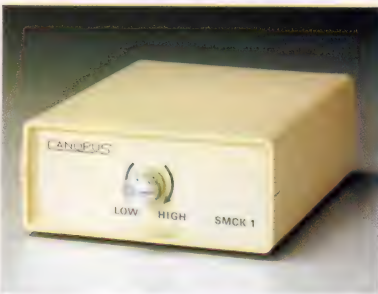
●PC-9801E、F、M、VF、VM、VM21、VX、UX、CV、UV、RX、RA、ES、EX(ただし、PC-9801E、F、M、VFは44.1kHz、48kHzのサンプリングクロック時1チャネルしか使用できません。)

●メモリ	本体640Kバイト以上・漢字ROM
バンクメモリ/1Mバイト以上必要(I/Oバンク方式)	
●マウス	NEC製または同等品
●OS	MS-DOS Ver.2.0以上(NEC製)

〈オプション〉

外部クロックユニット(連続可変型)SMCK-1　¥9,800

サンプリング・クロックを5kHz～50kHzの間で連続可変できる外付ユニットです。

●型番 HAB-1130-01

FLASH-16(DSPボード)　¥298,000

FFT・ディジタルフィルターを高速演算します。

●型番 CAU-1101-01-13××

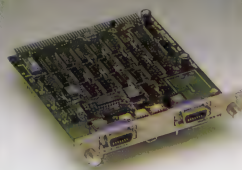
ASIP(プログラマブルゲイン、フィルタユニット)　¥198,000

●型番 H型 HFU-7201-01H　L型 HFU-7203-01L

●接続台数	最大16ユニット(32チャネル)
●入力ゲイン	×1、×5、×10、×50
●ローパスフィルタカットオフ周波数	
H型	20k、10k、5k、2k、1k、500、200、100(Hz)
L型	2k、1k、500、200、100、50、20、10(Hz)
●ハイパスフィルタ	0.05(Hz)

カノープス電子(株)
本社〒658神戸市東灘区西岡本1丁目4-30カノープスビル/ご注文・納期のお問い合わせは〈営業部〉TEL:078(411)5292　Fax:078(431)7610
技術的なお問い合わせは〈テクニカルインフォメーション〉TEL:078(412)7166(月曜～金曜/PM1:00～4:00)　Fax:078(411)5084

●表示価格に消費税は含まれておりません。●MS-DOSは米国マイクロソフト社の商標です。

リレー・カウンタ・PMC・DAC

ロータリーエンコーダ用カウンタ

ロータリーエンコーダ(2相信号)専用の2CH、16BITアップダウン・カウンタです。

- 1～4逓倍可能、追従速度100KHz
- 16BITバイナリ、2CH独立制御、プリセット可能
- フォトカプラ絶縁入力(耐圧AC500V)
- 入力回路、エンコーダ共用DC-DC搭載可能

UDC-4298CPC ¥49,000

メモリ・DMA機能付 2CHユニバーサル カウンタ

1024データ分の FIFO 型バッファメモリ内蔵、DMA機能付の2チャンネル・16BITユニバーサルカウンタです。

- 周波数　：2CH同時、連続測定
- 周期　：1CHのみ、連続測定
- 積算計数：1CHのみ、連続測定
- パルス幅：1CHのみ、連続測定
- 時間々隔：両CH入力間、連続測定

追従速度：1MHz
時間分解能：1μs
各測定サイクル間のミスカウントはありません

UCM-4398BPC ¥145,000

2CH、アイソレーション12BIT DA

2チャンネル12BIT・DAコンバータをパソコン側とフォトカプラで絶縁したものです。電圧出力型と電流出力型があります。

- 標準0～＋10V出力(±10V、±5V可能)
- 電流出力型は4～20mA
- セトリング・タイム 20μs
- 耐圧AC500V(1分間)

電圧出力 TDA-2598XPC-V ¥85,000
電流出力 TDA-2598XPC-I ¥98,000

2CH、アイソレーション16BIT DA

2チャンネル16BIT・DAコンバータをパソコン側とフォトカプラで絶縁したものです。プログラマブル割込みタイマが付属しているので、任意の波形発生器として使用することもできます。

- 標準0～+10V出力(0～+5V、±5V、±10V可能)
- セトリング・タイム 15μs
- 耐圧AC500V(1分間)

TDA-2698XPC ¥128,000

24CH機械接点リレー

汎用24接点(標準実装8接点)リードリレー出力ボードです。用途によっては水銀リレーに交換して使用することもできます。

- 使用リレー：サンユー製S-105N 定格100V 0.2A

水銀リレーに変更も可能

8CH実装 SWR-6398BPC- 8 ¥44,000
16CH実装 SWR-6398BPC-16 ¥54,000
24CH実装 SWR-6398BPC-24 ¥64,000

接点入力&リレー(SSR)出力

機械接点の読み込み、および電力制御スイッチ接点が混載されています。DC用には機械接点リレー、AC用にはSSRが最適です。

- 8接点入力(フォトカプラ絶縁/DC-DC電源付)
- 4リレー(5A)、または4SSR(2A)出力

SSR出力 SWR-6198BPC ¥46,000
リレー出力 SWR-6298BPC ¥45,000

2CH パルスモータ制御

専用LSIを使用しているので簡単な制御コマンドを与えるだけです。また、最大1A(相電流)のドライブ能力があるので小型のモータなら直接駆動することができます。

- 3～5相モータに適用
- 最高速度5kpps
- 定速、加減速、シングルステップ、基準点まで移動、etc.
- トピックス 全入出力アイソレーション、実用的ソフト付

PMC-7098BPC ¥82,000

2CH パルスモータ制御

専用LSIを使用しているので簡単な制御コマンドを与えるだけです。市販のパルスモータ・ドライバ、またはDCモータ・ドライバと併せて使用します。

- 定速、加減速、基準点まで移動、シングルステップ等、最高速度10kpps
- フォトカプラ駆動用電源(DC-DC)内蔵
- トピックス 実用的ソフト付

PMC-7198BPC ¥74,000

32BIT絶縁出力 DIO

8BIT×4ポートのフォトカプラ絶縁出力ボードです。出力はオープンコレクタ、またはTTLレベルを選択することができます。

- 耐圧AC500V(1分間)
- 駆動電流max200mA
- TTL出力用DC-DCコンバータ内蔵

DIO-3398BPC ¥46,000

32BIT絶縁入力 DIO

8BIT×4ポートのフォトカプラ絶縁入力ボードです。接点読込みに最適、また各ポートごとにストローブ(ラッチ)入力として使用することもできます。

- 耐圧AC500V(1分間)
- ストローブ信号による割込み可能
- 入力回路駆動用DC-Dコンバータ内蔵

DIO-3498BPC ¥56,000

アイソレーション DIO

フォトカプラ絶縁の16BIT入力、16BIT出力です。

- 16BITラッチ出力(フォトカプラ→オープンコレクタ)
- 16BITバッファ入力(フォトカプラ電流駆動)
- 入力BITの2本は割込みにも使用可能

DIO-2298BPC ¥36,000

TTL入力・オープンコレクタ出力 DIO

24BIT(8×3ポート)TTL入力、16BIT(8×2ポート)オープンコレクタ出力の汎用デジタル入出力です。

- リレー等を直接制御できるオープンコレクタ出力(max.40mA)
- 24BIT入力ポートは割込みにも使用可能(DIP-SW指定)

DIO-3298CPC ¥25,000

当社情報提供パソコン通信 MICRO-FALA を運用中です。ゲストは製品仕様書、サンプルソフト等を直接入手できます。通信パラメータはPC-VAN同様(300～2400BPS)です。当面の運用は月～金曜の10～19時、TEL.03(247)1239で行います。

☆マーク付の製品にはフロッピーが付属します。
☆全製品にBASIC/機械語のリストが付属します。

アナログ入力 for 9801

時代が求めた **DMA,メモリ付 高速AD** これが新標準

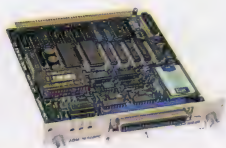
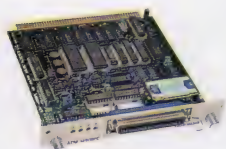
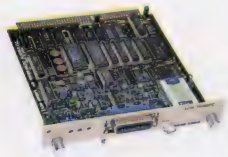
8CH同時サンプリング 12BIT高速ADコンバータ
- 変換速度：5μs＋5μs/CHサンプル
- 入力範囲：±5、±10、0～＋5、0～＋10
- 入力形式：8CHシングルエンド
- 非直線性：0.05%FS

ADM-1998BPC ￥210,000

16CH(差動8CH) 12BIT高速ADコンバータ
- 変換速度：5μs/データ(CH)サンプル
- 入力範囲：±5、±10、0～＋5、0～＋10
- 入力形式：16CH(または8CH差動)
- 非直線性：0.05%FS

ADM-1498BPC ￥198,000

16CH(差動8CH) 16BIT高速ADコンバータ
- 変換速度：25μs/データ(CH)サンプル
- 入力範囲：±5、±10、0～＋5、0～＋10
- 入力形式：16CH(または8CH差動)
- 非直線性：

ADM-1698BPC ￥220,000

ここに御招介した高性能ADコンバータのほかに低価格・汎用機もございます。

★ハードウェアによる自動サンプリング
指定された条件でハード的に自動サンプリングを行いますから、ソフトは簡単、高度なテクニックは不要です。

★1024データ分のバッファメモリ付
サンプリング(AD変換)されたデータはFIFO型の入出力非同期バッファメモリに転送されます。

★高速DMA転送も可能
バッファメモリからのデータ読出しはI/Oポートのほか DMAも可能この動作はADサンプリングより速ければよく、同期をとる必要がありません。

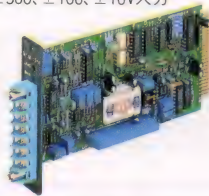
★プログラマブル・クロック内蔵
内部クロックは250ns刻みで任意の値を指定することができます。また外部クロックによる動作も可能です。

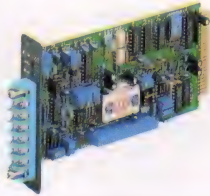
★プログラマブル・トリガ回路内蔵
アナログ入力CHφの信号が、指定レベルを越えたときにADサンプリングを開始することもできます。 また、外部トリガも可能です。

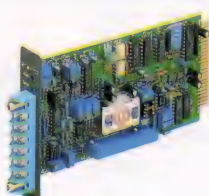
★実用的サンプルプログラム付
ADサンプリングからデータのグラフ表示、ファイル作成、等の機能があります。基本機能はサブルーチン化されており、ユーザプログラムの追加記述も容易です。(N88版& MS-DOS版)

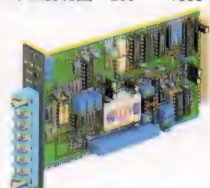
センサ直結！ **絶縁型センサアンプ** これで完璧！

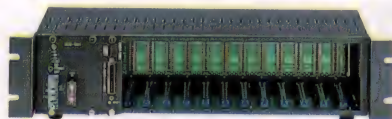
MCM-1201 (低レベル用)
- ±0.1、±1、±10V入力

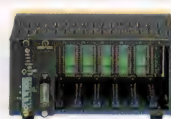
MCM-1202 (高レベル用)
- ±500、±100、±10V入力

汎用電圧入力

MCM-1102
- 圧力ゲージ・ブリッジ(差動)入力
- ±10、±1、±0.1V入力範囲

PSゲージ入力

MCM-1103
- ロードセル・ブリッジ(差動)入力
- ±10、±0.15、±0.1V入力範囲

ロードセル入力

MCM-1101-K CA熱電対用
MCM-1102-J IC熱電対用
- 冷接点補償付
- 入力温度範囲－200～＋1000℃

熱電対入力

(共通仕様) ●出力±10V ●セトリングタイム100ms ●温度ドリフト50μV/℃ ●周波数特性 fc＝約15Hz ●絶縁耐圧AC500V ●電源±15V80mA ●寸法 24.5W×99H×162D ●使用温度 0～＋45℃ ●付属品：コネクタ、圧着端子 ●オプション：4～20mA電流出力

電源ボックスも用意しました。

(12スロット用) ←BOX-1600　(5スロット用) BOX-0800→

オプション・ソフト(資料を御請求下さい)
- 各ADボード用FFT解析(N88-BASIC/MS-DOS)
- 各ボード用ライブラリ(C言語)

マイクロサイエンス株式会社
〒167 東京都杉並区西荻北3丁目2番3号 TEL 03(396)8362代表

資料請求No.31

NECパーソナルコンピュータ PC-9800シリーズ用

イーサネットボード

FA-LANⅢ(98)H-01 -02

● Ether/Thin Wire両用タイプ H-01 ¥138,000
● Ethernetタイプ H-02 ¥116,000
(別途通信ソフトが必要です。)

ASIC採用により低消費電力化、RAMを512KB搭載することにより転送スピードを3倍にアップ。TCP/IPプロトコルに加えMS-NETWORKSをサポート、パソコン同士のネットワークも実現。さらにinformix NETをサポートするなどハード、ソフトともより使い易くなりました。

●先進のハードウェア

ASIC採用によりコンパクト化、低消費電力化を実現。

コンパクト化することによりEther/Thin Wire両方共1枚のボードに搭載、拡張や変更への対応も万全です。消費電力も2000mAから1300mAと大幅低下、外部電源をなくしました。

転送速度大幅アップ、回線数も16回線に増加。

RAMを512KB搭載することにより、大きなファイルも1度に取り込むことができ、転送速度が当社比3倍に向上(70KByte/sec)。また同時に交信できる回線数が8から16回線に倍増。16台のパソコンと同時に送受信が可能になりました。

●豊富なサポートソフト

MS-NETWORKSをサポート。

パソコンだけでネットワークを構築したいと思われる方にとって有効なネットワークを提供。市販の多彩なアプリケーションソフトが利用できます。

informix NETをサポート。

社内のデータベースを構築したい方のために、アスキー製informix NETをサポートしました。本格的分散処理環境でのデータマネジメントシステムが構築できます。

■FA-LANⅢ(98)H関連ソフト

● FTP・TELNET ¥30,000
● MS-NETWORKS ¥20,000
● PC-NFS ¥90,000
● ソケットインターフェイス ¥120,000
● FTPサーバ ¥86,000
● バスインターフェイスマニュアル ¥2,000

● 富士通FMR50/60/70及びIBM-PC/AT、TOSHIBA AX用 同時発売中!!

※informixは米国インフォミックスソフトウェア社の登録商標です。
※PC-NFSは米国サンマイクロシステムズ社の登録商標です。

● パソコンネットワーク無料実習コース〈於 東京〉
毎月第3水曜日

当社製TCP/IPプロトコル搭載《FA-LANⅢ(98)》の機能を中心にネットワークの概要をご紹介し、その後イーサネットを用いてEWSと接続されたパソコンを操作、実習していただきます。

資料請求No.32

新登場

MS-DOSが走る!!

98バスボードコンピュータ

CPUボードにパーソナルコンピュータと同じMS-DOSを移植。アプリケーションプログラムの開発は低価格なパーソナルコンピュータを用い、しかも高級言語の使用により、ソフト生産性を飛躍的に高めることができます。またボードサイズは、NEC PC-9800シリーズの拡張スロット仕様であるため、100種類以上の当社製PC-MODULEや市販のボードが利用でき、自在にシステム構築が可能です

パーソナルコンピュータで開発した アプリケーションソフトがそのまま走ります。

CPUボード・PC-80286(98)ボード上にMS-DOS Ver3.3をROMで搭載。従って、パーソナルコンピュータ(MS-DOS環境下)で開発した実行形式のアプリケーションソフトが直ちに走ります。(但しグラフィック関連除く)

用途に合わせて自由なシステム構築が ローコストで可能。

CPUボード・PC-80286(98)はPC-9800シリーズ拡張スロット仕様のため、100種類以上の当社製PC-MODULEや市販のボードが利用でき、極めて多様なシステム構築が可能です。

周辺機器との拡張性、信頼性を向上。

CPUボード以外にプリンタ、キーボード、CRT、FD/HD等周辺装置との接続や、操作性、信頼性を高めるRAS機能ボードを用意し、さまざまなシステムの拡張に応えます。

リアルタイムOSもサポート。

MS-DOS以外にFA・LA用途で必要とされる当社製リアルタイム・マルチタスクの動作環境をサポートします。(開発中)

● リアルタイム-BASIC(98) ● リアルタイム-MS-DOS(98) ● リアルタイム-iRMX(98)

■98バス周辺関連ボード

品名	価格	品名	価格
● CPUボード(MS-DOS搭載) **PC-80286(98)M**	¥148,000	● プリンタ/RS-232Cボード **PC-COM(98)**	¥58,000
● ROM-DISKボード **PC-ROM(98)**	¥38,000	● RAS機能ボード **PC-RAS(98)**	¥42,000
PC-CRT(98)	¥68,000	**PC-FD/SC(98)**	¥48,000

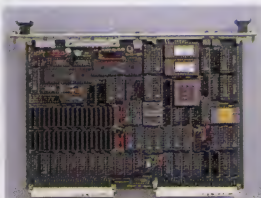
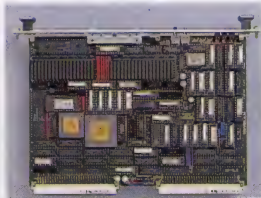
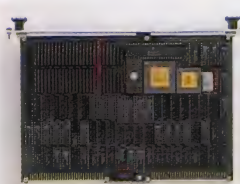
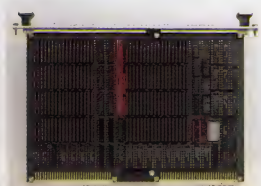
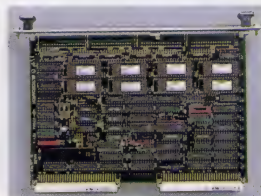
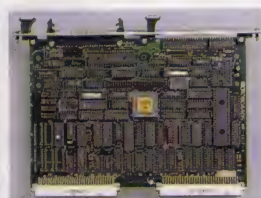
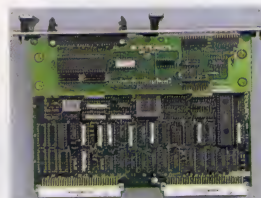
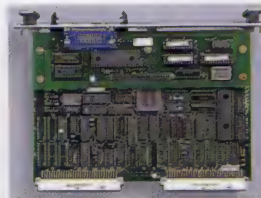
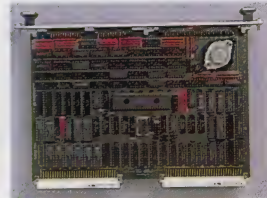
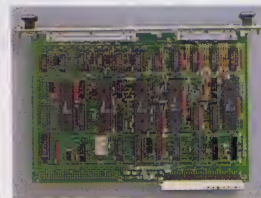
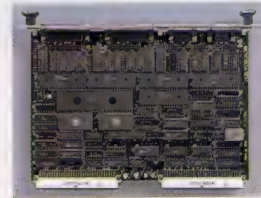
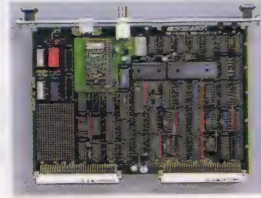
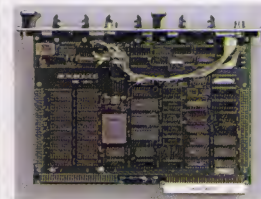
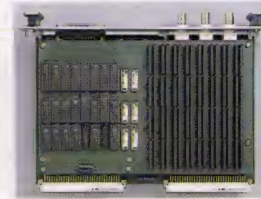
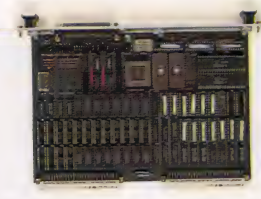
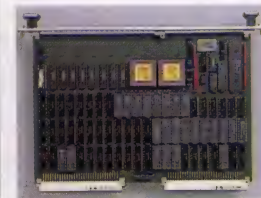
※MS-DOSは米国マイクロソフト社の登録商標です。

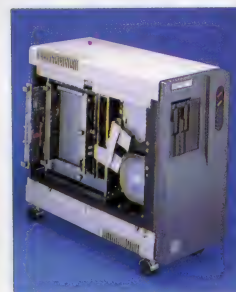
コンテックFAマイコンセンター
〒105 東京都港区芝2-29-11
TEL(03)769-1061

NEC マイコンショップ
コンテック マイコンセンター
〒555 大阪市西淀川区姫里3-9-31
TEL(06)472-0265

全ての32bit VMEシステム・ビルダーへ捧げます。

HPV Series

VMEバス

MB-2110　68000CPU

MB-2150　68020CPU(16MHz)

MB-2170　68020CPU(25MNz)

MB-2240　16MB DRAM(4／8M)

MB-2280　EPROM／SRAM

MB-2310　FDC／HDC

MB-2312　SCSIアダプター

MB-2313　GPIBアダプター

MB-2370　SMD I／F

MB-2410　シリアル 6ch
セントロニクス 2ch

MB-2450　シリアル 8ch
インテリジェントI／O

MB-2470　OS-9／LAN

MB-2510　グラフィックス

MB-2520　高解像度グラフィックス

MB-2530　レーザビーム・プリンタ
コントローラ

VMEシステム・キャビネット

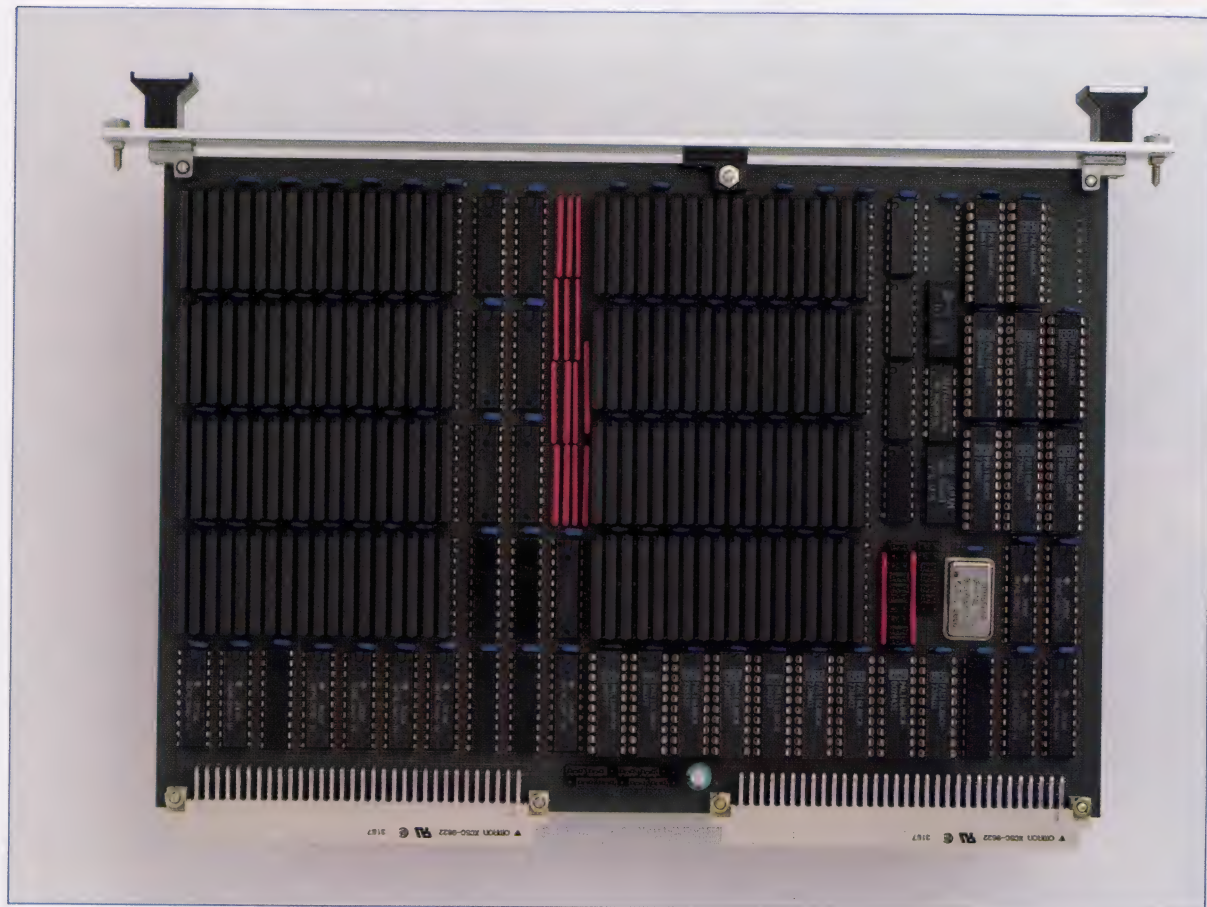
MB-2710　DSP

HPVシリーズ(High Performance VME board series)のボードは､お客様より性能で支持を受けています。それは、性能を引き出すOS-9ソフトウェアでサポートされているからです。OS-9/68020のソフトウェアは、システムビルダーが容易にシステムを構築できる環境が用意されているからです。性能とは、ハードウェア仕様ではなく、ソフトウェア環境を含めたシステムにおける性能です。
HPVシリーズは、それらを追求した結晶です。

VME busボード及びOS-9システムのご用命は㈱マイクロボードへ

資料請求No.34

今月の推薦製品

MB-2240

4/8/16MB DRAMボード

◆特徴◆

◆メモリー容量は4MB、8MB、16MBの3タイプが有ります。
◆拡張バスであるVSBバス（The VMB Subsystem Bus）を採用しています。
◆デュアルポート構成ですので、VMEバス、VSBバスのどちらからもアクセスできます。
◆データバスは32ビット構成ですので、32ビット、16ビットのどちらのシステムでも対応できます。
◆アドレスは32ビット（拡張）、24ビット（標準）に対応できます。
◆パリティチェックによるエラー検出機能付きです。
◆アドレス・モディファイアは、スーパーバイザ・プログラム・アクセス、スーパーバイザ・データ・アクセス、ユーザ・プログラム・アクセス、ユーザ・データ・アクセスの有効／無効の設定が可能です。
◆VMEバス規格 Rev.C準拠
◆VSBバス規格 Rev.A準拠

販売代理店

東京都千代田区外神田1-5-8
〒101 phone.(03)251-1701

〒273千葉県船橋市本町4-41-19本町セントラルビル3F
PHONE:0474-22-1741(代表) FAX:0474-22-1759

AD	高機能・高性能AD 特長&特徴	AZI-275	98AD12(32/16)-PTC	¥138,000
		タイマ内蔵シングル32点差動16点バス絶縁型 12ビットA/D変換ボード		

■外部入力端子装備（外部トリガ入力/汎用入力）
外部信号(TTLレベルのパルス信号)をジャンパSWの設定により外部トリガ入力と汎用入力機能の使用選択が行えます。
- 外部トリガ入力機能：外部トリガ入力信号をCPU(ホスト)用割り込み、もしくはA/D変換スタート信号として使用できます。
- 汎用入力機能：汎用入力(TTLレベル)としてスイッチ等の取り込みができます。

■アナログ入力保護回路内蔵
入力保護回路マルチプレクサを採用しておりますのでアナログ信号入力電圧の有無に関係なくボード(98BUS)側の電源をON/OFFしても大丈夫です。
- POWER ON時：±35V(MAX)
- POWER OFF時：±20V(MAX)

■汎用出力端子装備
外部装置等への汎用出力(TTLレベル)をコントロール信号として使用できます。

■外部トリガ出力端子装備
外部機器との同期運転用としてA/D変換スタート信号(TTLレベル)を出力できます。

■アンフェノールコネクタ装備
ソルダータイプの適合ソケットを標準添付していますので用途に合わせてフレキシブルな配線が可能です。

■豊富な入力電圧範囲
入力電圧範囲は0～5V、0～10V、±2.5V、±5V、±10Vとジャンパ SWの設定により選択が可能です。さらに入力ゲインを×1倍、×10倍×100倍と3段階を任意に設定が行えます。

■基準電圧出力端子装備
コネクタ端子より5.000Vの基準電圧を出力していますのでこれを利用することにより外部電源を使用しないで入力回路の調整が行えます。

■電源出力端子装備
コネクタ端子より外部へ電源電圧を出力していますので用途に合わせて使用できます。(+15V(5mA)、−15V(5mA))

■EMIフィルタ搭載
アナログ信号入力部にEMI除去フィルタを設けており外部ノイズに対して効果的な除去を行う様に設計されています。

新発売

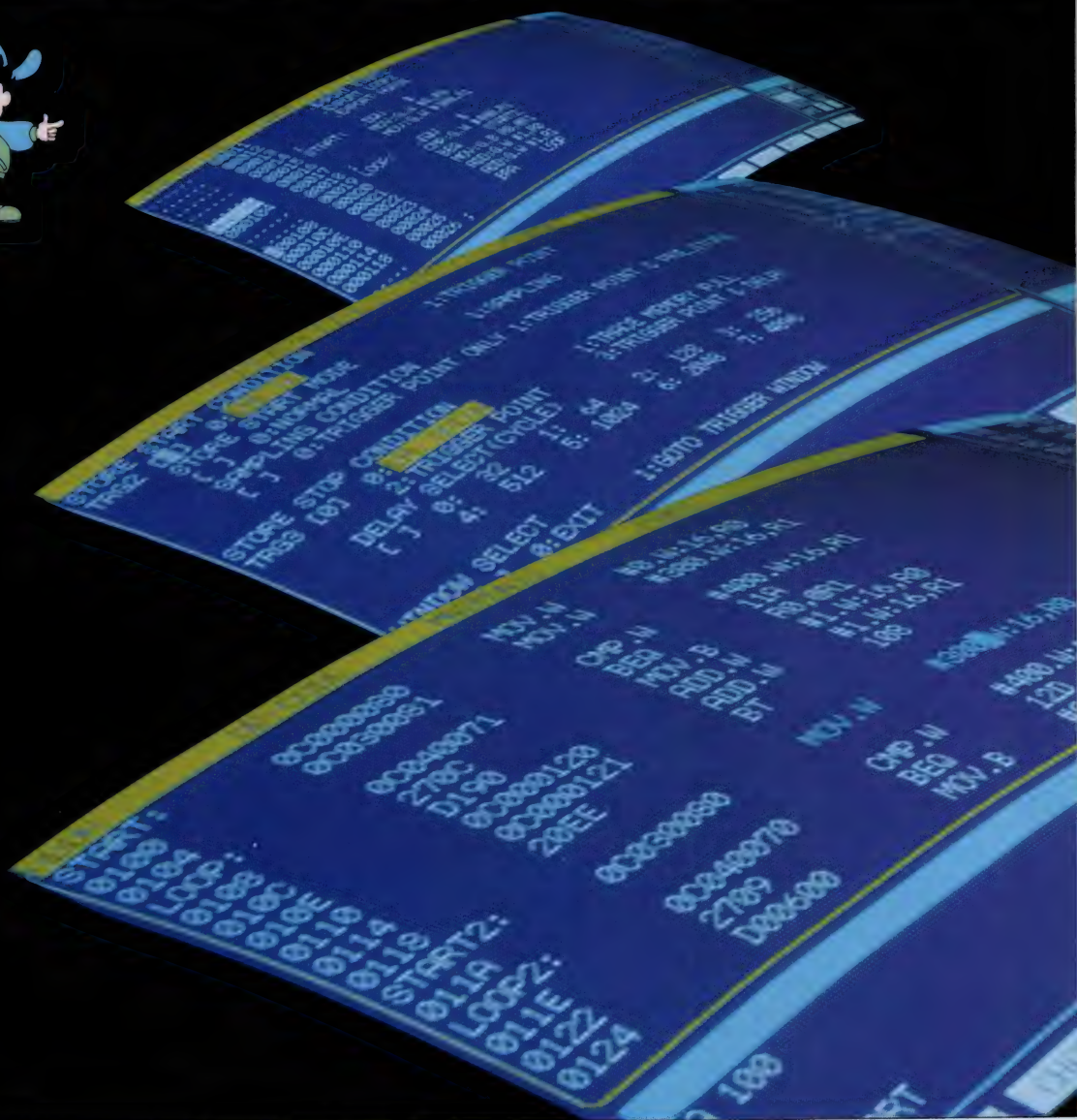
■サンプリング・クロック(タイマ)内蔵
プログラマブルタイマ(μPD71054C相当品)を採用、割り込み発生又は、A/D変換スタート信号として使用できます。サンプリングのタイミングを決めるクロックとして内部及び外部クロックの選択が可能です。サンプリングタイマにサンプリング周期をソフト的にセットするだけで後は自動的にセットした周期でサンプリングを行います。
- インターバルタイマ設定範囲　：4μsec～4296sec(71.6分)
- A/D変換スタート周期設定範囲：30μsec～4296sec(　〃　)

■サンプルプログラム例（各種言語サポート）
BASIC、ASM、C言語の各種言語によるサンプルプログラム例を取扱説明書に掲載しておりますので用途に合わせてご利用いただけます。

■固定チャンネル時、変換時間の高速化を実現
チャンネル切り換えとA/D変換スタートの制御を独立に行うことができますので固定チャンネル(1chのみ)使用時の変換時間の高速化を計ることができます。
- 変換速度　チャンネル切換時　65μsec
　　　　　　チャンネル固定時　30μsec

■割り込み機能
A/D変換終了割り込み、インターバルタイマ割り込み、外部トリガ割り込み信号を発生することが可能で、割り込み発生信号及び割り込みレベルはジャンパSWにて任意に設定可能です。

■フォトカプラ絶縁
アナログ信号にデジタル信号が混入しない様に、又アナログ部の異常電圧がCPU(ホスト)側に影響をあたえない様に高速フォトカプラで電気的に絶縁(分離)しております。その上多層基板を採用しておりますのでノイズに強くいつでも正確なデータをサンプリングできる様になっております。

■ロータリースイッチ採用（I/Oポートアドレス設定）
ロータリースイッチ採用により、16進(HEX)そのままの設定が可能で従来のディップスイッチの様な設定ミスから解消され開発もスムースに進められます。又16ビットフルデコード設定可能により64KのI/O空間に思いのままに割り付けが可能です。

●高速・高機能、そして低価格…専用ICEならではのメリットです

ことし登場した日立の8ビットMPU《H8/532》は、アドレス空間最大1MBという大容量と汎用16ビットMPUをしのぐ高速性から早くも引く手あまたの状態、──その専用ソースコードデバッガとして誕生した「えみゅっ太くん」も、汎用高級ICEの高速・高機能をすべてカバーした上、本体構成85万円という低価格を実現し、大好評をいただいています。ホストはユーザお手持ちのPC-9801。フルスクリーン感覚で実機デバッグが行えるマルチウィンドウ型ソースコードデバッガをはじめ、プログラム実行中、メモリ内容の変化まで動的に表示する、EST（エクゼキューション・スチール）機能や、UMAC（ユーザ・メモリ・アクセス・コントローラ）など、強力なハードウェア、ソフトウェアの支援により、理想的なリアルタイムデバッグを実現。さらに、専用機ならではの操作性、習得性もユーザに好評です。

●フルスクリーンを活用した強力なダイレクト編集機能

「えみゅっ太くん」は、マルチウィンドウ型ソースコードデバッガを標準装備。ウィンドウ内のどこにでもカーソルを直接持ち込んで、すばやく、的確に作業が行えます。コマンドラインの入力データは、1ライン128文字まで挿入、削除が可能。ダイレクト編集機能により、メモリウィンドウ レジストウィンドウ等に直接、カーソルを持ち込んでの内容変更はもちろん、逆アセンブル画面上で直接ニーモニックを変更しラインアセンブルすることさえ可能になりました。
ヒストリ機能も強力で、コマンドラインから入力したコマンドは最高64個前まで保存され、スクロールの巻戻しにより、再実行が可能です。また、エミュレーション実行中でも、MS-DOSコマンドが実行できます。さらに、画面が任意に変えられるオンラインヘルプ機能も充実。「えみゅっ太くん」は、まさにフルスクリーン感覚の強力なダイレクト編集機能を持つ開発支援機といえます。

●メモリ内部まで動的に実行状況が表示・確認できるEST機能搭載

高速・高機能をユーザオリエンテッドなかたちで追求した「えみゅっ太くん」。EST（エクゼキューション・スチール）機能と、独自開発のUMAC（ユーザ・メモリ・アクセス・コントローラ）が一体となり、高速アクセスを実現しています。さらに、実行状況はメモリウィンドウで動的に表示・確認できます。もちろん、

誕生!!

SILICONTOOL PROFESSIONAL ICE

えみゅっ太くん H8/532

さすが、H8〈専用〉ICE。

フルスクリーン感覚の実機デバッグを、ここに実現!

ハードウェアブレーク、リアルタイムトレースなどの機能も強力。ハードウェアブレークは、CPUバスデータの組み合わせができるトリガーポイントが4ポイント使用可能で、さらにシーケンシャルブレークも可能。トレースメモリは8192サイクル×72ビット。また、エミュレーションメモリ256KBのほか、内蔵ROM、RAMへの専用代替メモリを持ち、リアルタイム性を損うことなく内蔵のROM、RAMのエミュレーションが可能です。「えみゅっ太くん」は、クロック最大周波数10MHzの範囲で、ユーザ待望の、透過性の高いエミュレーションを実現します!

H8/532 専用ICE

『えみゅっ太くん』

標準価格 850,000円(消費税別)

《本体構成》

- 本体
- PC-9801用インターフェースボード
- PC-9801用ホストケーブル
- H8/532 CPUプローブケーブル
- 外部プローブケーブル
- マルチウィンドウ型ソースコードデバッガソフト

《オプション》

- アセンブラ(ADC版)* 標準価格95,000円(消費税別)

※VAX版日立純正アセンブラを、ライセンス契約にもとづき、ADCがMS-DOS版に移植し、タグジャンプ機能、リストファイルへのエラーメッセージ表示機能を追加しました。

- Cコンパイラは日立製作所製のものが利用できます。

*「シリコンツール」および「えみゅっ太くん」はADCの商標です。
*標準価格に消費税は含まれておりません。

NIPPON CHEMI-CON ケミハン株式会社

開発営業部/東京都品川区東五反田3-14-13 Tel:03(448)1251 Fax:03(448)1277

資料請求No.39

流れはSCSI&大容量!

SCシリーズ 7タイプ 衝撃的にデビュー!

NEC PC-9801-55上位コンパチ
SCSI I/F対応!

最大 600MB!

最高 16m sec.以下!

光磁気ディスク同時新発売!

PC-9801/PC-286/Macintosh/NEWS用
SCSI I/F対応ハードディスクシステム　新発売　8月25日発売予定!!（NEWS用は9月21日発売予定）

CA-6016SC 600MB 16m sec.以下 定価¥948,000
CA-3016SC 300MB 16m sec.以下 定価¥588,000

PC-9801/PC-286/Macintosh/NEWS用
SCSI I/F対応光磁気ディスクシステム　新発売

CA-6080MO 600MB 80m sec.以下 定価¥480,000（メディア1枚付）
別売メディア¥30,000

PC-9801/PC-286/Macintosh用
SCSI I/F対応ミニタイプハードディスクシステム　新発売　8月25日発売予定!!

CA-2020SC 200MB 20m sec.以下 定価¥378,000
CA-1025SC 100MB 25m sec.以下 定価¥218,000
CA-0818SC 80MB 18m sec.以下 定価¥188,000
CA-0428SC 40MB 28m sec.以下 定価¥128,000
＊CA-0428SCはMacintosh＋のみ使用できません。

NEC PC-9801-55上位コンパチブル・SCSI I/Fボード **PC-98M20** 定価¥38,000

NEC PC-9801-55バージョンアップサービス ¥5,000+送料
＊PC-9801-55を上位コンパチブルのPC-98M20仕様に機能アップします。詳しくは当社までお問合わせください。

★世界的に流れはSCSI/

これからのパソコンI/Fの主流はSCSI。キャラベルのSCシリーズは、全機種OSを無視しない「本物」のSCSI仕様。今、最も新しいスタイルのHDラインアップ、衝撃的にデビュー!

★PC-9801-55と完全コンパチのSCSI仕様/ SCSI I/Fボードは、単なるハードディスクI/Fではありません/

PC-9801-55のすべての機能＋αのSCSI I/Fボード「PC-98M20」は、EPSON PC-286シリーズにも使用できます。このボードの使用で、NEC純正HDとの混在使用が可能。また、NEC純正のSCSI仕様HDをEPSON PC-286シリーズに接続することも可能です。SCSI I/Fボードは、単なるハードディスクI/Fではありません。1枚あれば、後はデイジーチェーンで各種SCSI機器が接続できます。よってPC-98M20は別売とし、その分システムの価格をおさえました。

★デバイスドライバー不要/ OSを無視しない「本物」のSCSI指向/

MS-DOS Ver3.3/3.3A, OS/2 Ver1.0以上, PC UXリリース3.0A以上など、デバイスドライバーを使用しないで、通常のフォーマットで使用できます。

★最大600MB/ 豊富な7タイプ/

低価格の40MBから余裕の600MBまで、光ディスクを含めニーズによって選べる7タイプを用意しました。

★光磁気ディスクも新登場/

バックアップ用として有効な光磁気ディスク。ストリーマテープに比べ非常にスピーディなバックアップが可能になります。1枚両面で600MBの大容量を誇る画期的ニューメディアです。リードライトが可能であるため、ハードディスクと同様の使い方もできます。

★最大1パーテーション128MB/ 最大2.4GBまで増設OK/

分割は、最小1MBから最大128MBまで自由に設定できます。また、デイジーチェーンにより最大2.4GBまで増設可能。ハードディスク(光磁気ディスクを含む)なら4台、他のSCSI機器と合わせて計7台まで接続できます。しかも、どのHDからでもブートアップ可能。

★万全のノイズ対策/

静電気等、外来ノイズに弱いスイッチングレギュレータは使用していません。ドロッパータイプの電源の採用により、ノイズによる誤作動を高レベルで防止します。

＊Macintosh, NEWSでご使用の場合、ケーブルは別売となります。

資料請求No.40

高性能FDDシステム同時新発売！

NEC PC-9801用フロッピーディスクシステム

CA-302FW 3.5″×2 定価¥67,800

CA-502FW 5″×2 定価¥67,800

●2DD/2HD選択可能
●ドライブセレクト(1,2または3,4)が自由に設定できます。
＊2HD、2DD用ケーブル別売

8月25日発売予定!!

新価格!! 新シリーズ登場でさらにお求め易くなりました。
高速、大容量、使い易くて信頼できるキャラベルのハードディスク群。

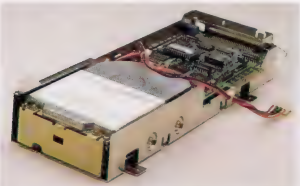
皆様のご愛顧に感謝をこめて 新価格 プライスダウン!!

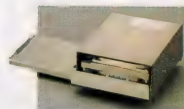
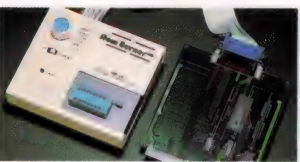
80MB単一でブートアップ!!
NEC PC-9800シリーズ用ミニタイプハードディスクシステム
CA-80LG 80MB 18m sec.以下 新価格¥218,000
●I/Fカード、ケーブル付

遂に達成!!40MBで18m sec.!!
NEC PC-9800シリーズ用ミニタイプハードディスクシステム
CA-44LG 40MB 18m sec.以下 新価格¥148,000
●I/Fカード、ケーブル付

よりお求め易くなりました!!
NEC PC-9800シリーズ用ミニタイプハードディスクシステム
CA-40LG 40MB 35m sec.以下 新価格¥118,000
●I/Fカード、ケーブル付

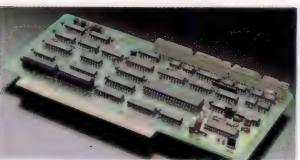
MS-DOS、新松etc.…ソフト内蔵!!
NEC PC-9800シリーズ用
ミニタイプハードディスク&ソフトウェアシステム
CA-80LGS 80MB 18m sec.以下 新価格¥258,000
●I/Fカード、ケーブル付
CA-40LGS 40MB 35m sec.以下 新価格¥168,000
●I/Fカード、ケーブル付

【搭載ソフトウェア】●日本語MS-DOS(Ver.3.1)アプリケーションソフト実行セット PS98-012-HMW一式〈日本電気㈱製〉●日本語ワードプロセッサ「新松」パッケージ一式〈㈱管理工学研究所製〉●アドレス帳(データ100名限定仕様)●グラフィックデータ集 ※ソフトウェアはすべて通常ご購入の場合と同様の保証、サービスが受けられます。

内蔵用も高速・高信頼性!!
NEC-9801RA2/RX2/VM11用内蔵ハードディスク
CA-818 80MB 18m sec.以下 新価格¥218,000
CA-428 40MB 28m sec.以下 新価格¥148,000

増設専用内蔵ハードディスク
CA-428＋(プラス) 40MB 28m sec.以下 定価¥138,000

Mac用も高速・高信頼性!!
Macintosh II/SE用
内蔵型ハードディスクシステム
EG-80II/SE 80MB 18m sec.以下 新価格¥188,000
EG-40II/SE 40MB 28m sec.以下 新価格¥128,000

NEC PC-9801用高性能ROMライター
Rom Burner
MR-01 本体価格¥38,500(ケーブル付)(電源を別途に必要)

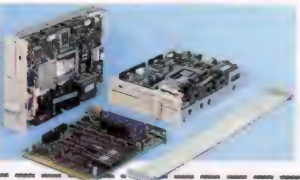
I/Fカード
PC-98M08 定価¥7,200

●EP-ROM2716から27512までのすべてのロムの読み書きが出来ます。●2764以上のロムは高速書込みが出来ます。(27128で約50秒) ●NEC9800シリーズ全機種対応します。●電源はDC9V〜12V用のアダプターを使用します。●インターフェースカードはPC98M08を使用します。●MS-DOS、CP/Mコントロールし、専用ソフトがつきます。

活用多彩、充実のI/Fカード群!!

98VFをVM仕様に!!
フロッピーマルチ I/Fカード
PC-98M17 定価¥39,800

●5″2DD、2HDの完全自動切替を実現しました。実際のVMと異なるのはメモリ容量(256KB)とクロック(8MHz)のみ、その他すべてVF仕様になります。8″バスにも対応。(ケーブル別売) FDDの交換は不要、ショートプラグの変更のみで使用できます。

9801F2, F3, Eを2HD/2DD自動切替に改造!!
8″内蔵2HD I/Fカード **PC-98M11markII** キット価格¥89,000

●2HD専用、2DD専用、2HD/2DD自動切替いずれの方法もスイッチによる切替が可能です。●8″および2HD I/Fカードで最大2HD2台、8″が2台接続出来ます。または2HDが4台接続出来ます。●2DD切替機能が内蔵されています。●VFOカードが内蔵されています。●PC-9801Fにこのカードを使用し、内蔵フロッピーを2HD、2DD両用のフロッピーに取り替えることにより、2HD/2DDが自動切替になります。

2HDフロッピー用VFOカード
PC-98M04markII 定価¥16,800

ストリーマ用I/Fカード
PC-98M16 定価¥5,000

2HD I/Fカード
PC-98M05 定価¥19,800

多機能HD I/Fカード
PC-98M03B 定価¥19,000

8″I/Fカード
PC-98M12 定価¥25,000

お求めは全国有名パソコンショップでどうぞ!
★NEWS用機種については㈱栄電子にご連絡ください。
★商品価格には消費税は含まれておりません。

開発・販売元 株式会社キャラベルデータシステム
Caravelle Data System
〒150 東京都渋谷区渋谷4-3-17-606 ☎03(498)5370㈹
神戸出張所/〒651 神戸市中央区雲井通4-1-11ラ・エリール207☎078(261)8170

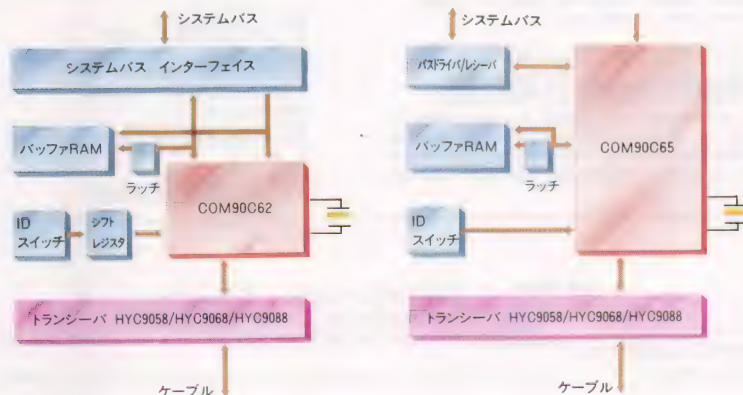
＊MS-DOS、OS/2は米国マイクロソフト社、新松は㈱管理工学研究所、Macintoshはアップルコンピュータ社の商標です。

TOYO MICROSYSTEMS CORPORATION

ARCNET®

日本のLANを決める!

ARCNET® LANコントローラ COM90C62 COM90C65

ARCNET®は、いまや、全世界で100万ノードを突破、現在、世界で導入されるパソコンLANの25%を占め、全米ではシェア・売上げ共にNo.2(1988年)を誇っています。また日本でも、信頼性とリアルタイム性を要求される各種制御用システムの通信系に使用され始めました。これは、ネットワークへのアクセスが混雑しても他のLANに比較して、最高のスループットを維持する実力が認められたからです。しかもコンパクトで低コスト開発が可能。日本におけるLANの標準として最適のLANシステムです。 東洋マイクロシステムズ㈱は、この度 米国SMC社による新しいARCNET®コントローラを発売しました。COM90C62は、従来のARCNET®コントローラとエンコーダ・デコーダ、およびクロック発振回路、パワーオンリセット回路をワンチップ化したものです。また、COM90C65は、さらにアドレスデコーダ等の周辺回路も内蔵。これらの製品によって、ハイ・コストパフォーマンスのLANハードウェアが容易に実現します。

《ARCNET方式LANの特長》

●アクセス方法/トークンパッシング・バス方式 ●ビットレート/2.5Mbps ●最大ノード数/255 ●トポロジー/スター、バス、ツリーおよびその混在 ●パケットデータ・サイズ/最大508バイト ●最大ケーブル延長/6.4km ●この他、最大の特長として、ネットワークへのノードの加入/離脱時あるいはトークン紛失時の、ネットワーク自動構築機能があり、COM90C62/COM90C65には、この機能を含んだ極めて簡略化されたARCNETプロトコルが完全に内蔵されています。

COM90C62は44pinPLCCまたは40pinDIPパッケージ、COM90C65は84pinPLCCパッケージ。

《その他のARCNETチップ製品》

品 名	内蔵機能	パッケージ
COM90C26	コントローラ	40 pin DIP/ 44 pin PLCC
COM90C32	エンコーダ/デコーダ	16 pin DIP
COM91C32	エンコーダ/デコーダ クロック発振器	16 pin DIP
HYC9058	同軸バス用 トランシーバ	20 pin ハイブリッド
HYC9068	同軸スター用 トランシーバ	20 pin ハイブリッド
HYC9088	ツイストペア線用 トランシーバ	20 pin ハイブリッド

※ARCNET®は、米国DATA POINT社の登録商標です。

販売代理店/㈱トーメンエレクトロニクス電子デバイス営業部門 ☎03-506-3673 インターニックス㈱マーケティング2部 ☎03-369-1105

SCSI

3つのバスでスループットが大幅向上。

高性能SCSIコントローラ MSD95C00

大容量の高速転送に最適

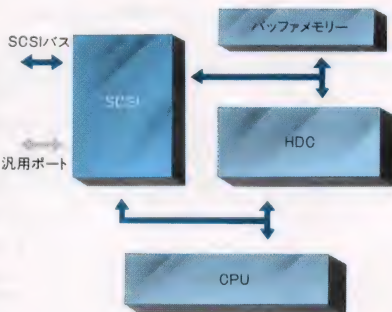
東洋マイクロシステムズ㈱は、この度米国SMC社のSCSIコントローラMSD95C00を新発売しました。このコントローラはANSI、X3T9.2に準拠し、イニシエータまたはターゲットとして機能するほか、ハード、ソフト共に以下の大きな特長を持っています。これにより、データ伝送速度も従来品に比べて大幅に高速化しました。

さらには、アプリケーションとしてはホスト用や各種プリンタ用、スキャナ用、ハードディスク用として最適です。

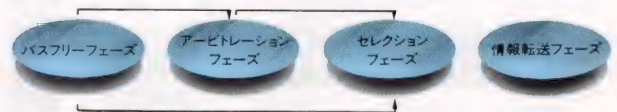
SCSI、DMA、CPUバスの3つのそれぞれ独立したチャンネルを持っている。

従来は、DMAとCPUバスを共用していたため、DMA中はCPUがHOLD状態となり何もできませんでした。

しかし、このMSD90C00のバスは図のように、SCSIバス、DMAバス、CPUバス共にそれぞれ独立しているので、DMA中でもリングバッファ中のデータの誤り訂正などが可能になり、その結果スループットの大幅な向上を実現しました。

また、8ビットの汎用ポートも内蔵しているので、周辺のチップの制御にも幅広く利用することが可能です。

《ソフトの特長》

コマンドの自動実行機能。

バスフリーフェーズ、アービトレーションフェーズ、セレクションフェーズを、レジスタの設定によりCPUの介在無しに自動実行することが可能です。バスフリーフェーズからアービトレーションフェーズを飛ばして、直接セレクションフェーズに入るなど設定可能です。

《性能一覧表》

プロセス	パッケージ	最大転送速度		オフセット値	アービトレーション機能
		非同期	同　期		
C-MOS	68 PIN PLCC	3M bps	5M bps	最大12	内　蔵

ドライバー	転送カウンタ	データーバッファ	発振周波数	データアクセスモード
不平衡内蔵	24ビット内蔵	12バイトFIFO内蔵	20MHz	バースト/シングルDMA

SMCと住友金属のジョイントベンチャー

東洋マイクロシステムズ株式会社

〒107 東京都港区赤坂4-9-17 赤坂第一ビル12F TEL.03-423-6650 FAX.03-423-6654

住商機電販売㈱電子第2部 ☎03-942-6771 丸紅㈱エレクトロニクスチーム ☎03-282-4146

最先端のディジタル/ビデオ技術

フォトロン IMAGE DIGITIZER®（イメージディジタイザ）は、NEC製PC-9800シリーズ用のパソコン画像処理装置です。ビデオカメラやVTRの映像をリアルタイムでメモリし、ビデオ モニタに表示するとともに 画像処理ソフトウェアにより多彩な画像処理を展開します。研究機関における顕微鏡画像解析、流体解析、レーザーパターン解析をはじめ、医療機関でのX線画像解析、製造部門における製品検査等、幅広くお使いいただけます。

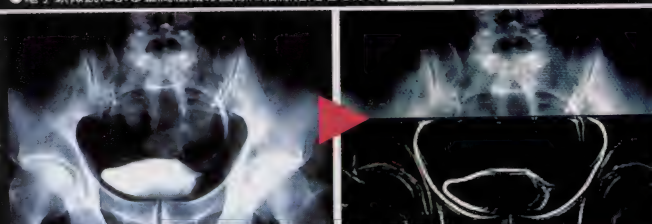
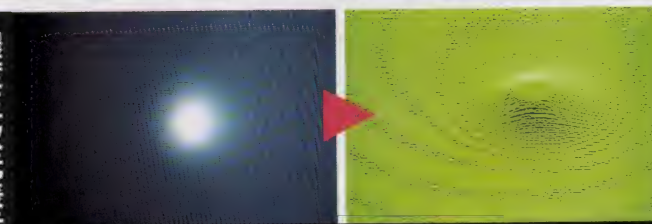
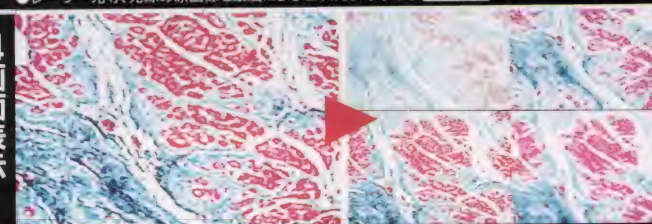
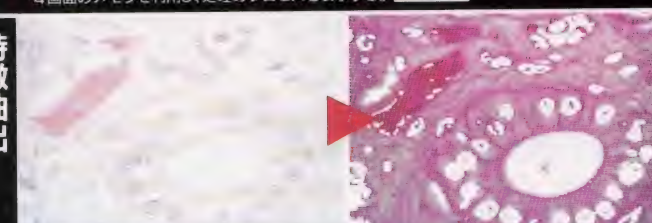
●電子顕微鏡による金属組織の画像に細線化処理を行う。FRM1-512

●レントゲン写真の原画像にエッジ検出フィルタを施す。FRM1-512

IMAGE DIGITIZER®

パソコン画像処理の決定版

●レーザー光等、光源の原画像を濃度により三次元表示する。FRM1-512

4画面表示

●組織切片の顕微鏡画像をFDM98-RGBにて処理、4画面のメモリを利用し、処理のプロセスを表示する。FDM98-RGB

特徴抽出

●顕微鏡画像を色成分により特徴抽出し、原画像に加算する。FDM98-RGB

ビデオカメラ

ビデオモニタ

イメージディジタイザ
FRM1-512
標準価格 ¥298,000
（画像処理ソフトウェア付）

イメージディジタイザ
FDM98-RGB
標準価格 ¥660,000
（画像処理ソフトウェア付）

イメージディジタイザ
FDM4-256
標準価格 ¥268,000

※価格には消費税は含まれておりません。
※IMAGE DIGITIZERは（株）フォトロンの登録商標です。

PC-9800シリーズ用 フォトロン・ビデオ画像処理ユニット
IMAGE DIGITIZER® シリーズ

	品名	型名	メモリ容量
ハード	イメージディジタイザ	FRM1-512	512×512画素×8ビット×1画面
	イメージディジタイザ	FDM4-256	256×256画素×8ビット×4画面
	イメージディジタイザ	FDM98-RGB	256×256画素×赤6×緑6×青6ビット×4画面
	品名	型名	備考
ソフト	アプリケーションソフト	FDM-Cライブラリ	C言語による画像処理開発用サブルーチンパッケージ
	アプリケーションソフト	FDM粒度パッケージ	粒子の面積、重心等の自動測定用ソフト

東京営業部 〒150 東京都渋谷区道玄坂2丁目8番7号（渋谷道玄坂ビル） ☎03(486)3471(代表) 画像システム課
大阪事業所 〒530 大阪市北区中之島5丁目3番101号（正宗中之島ビル） ☎06(444)2205(代表) 画像システム課

さらに充実したF² MC™-8のサポート体制で、大容量ソフトウェアの開発期間が大幅短縮。

富士通では、8ビットワンチップマイクロコントローラ(F²MC™-8)MB89700シリーズのサポートツールを大幅に機能拡張。C言語レベルデバッグ機能と、最適化Cコンパイラを新たに開発しました。これによって、プログラム開発言語(C言語)と同レベルでのデバッグが行え、開発言語とデバッグ時のセマンティックギャップが埋まり、開発効率が飛躍的に向上。またコード効率は従来のコンパイラに比べ約20%(当社比)もアップしました。プログラム開発から実機デバッグまでの一貫したC言語開発によって、ますます簡単に、短期間でソフトウェア開発ができるようになった、富士通のF² MC™プログラム開発支援システムを、ぜひご利用ください。

ソフトウエアサポート(C言語レベルでのデバッグをサポート)

- 最適化Cコンパイラ：各種最適化により、オブジェクトサイズは従来より約20%縮小しました。(当社比)
- ソフトウェアシミュレータ：実機システムデバッグ前にホストコンピュータのみによりソフトウェアデバッグが可能です。
- ホストエミュレータ：C言語レベルでのリアルタイムな実機システムデバッグを可能にしました。
- ライブラリマネージャ：相対形式のオブジェクトプログラムを編集・管理します。
- マクロアセンブラ
- リンカ
- スクリーンエディタ

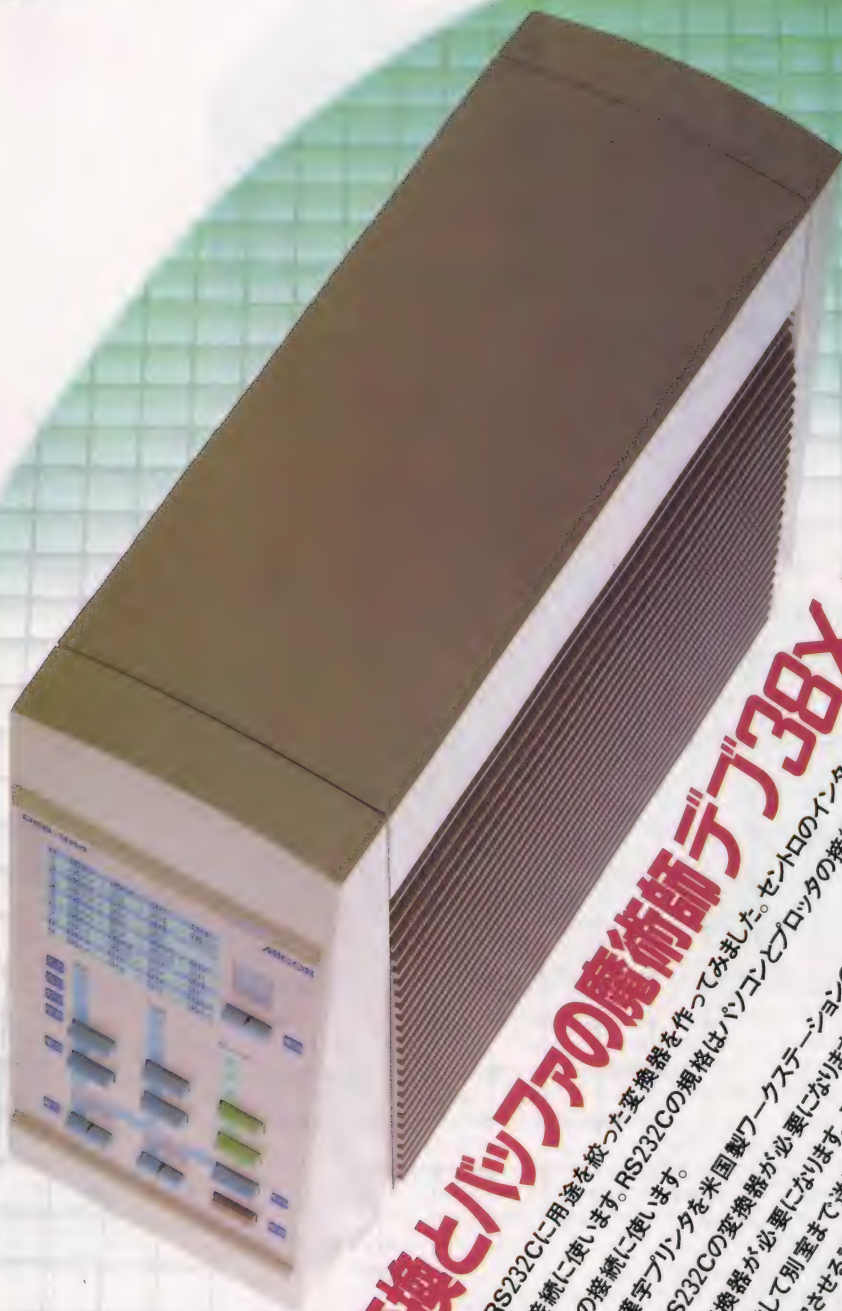
ハードウェアサポート

- MB2121(エバリュエーションツール本体) ● DUE89700ボード(ワンチップ用)
- DUE89702ボード(プロセッサ用) ● 拡張トレースボード

本広告に掲載の全商品ならびにそれに関連する消耗品等および役務について、ご購入の際、消費税が付加されますのでご承知おき願います。

「F²MC™プログラム開発支援システム」

富士通株式会社　電子デバイス第一統括営業部/電子デバイス第二統括営業部　〒100 東京都千代田区丸の内2-6-1（古河総合ビル） TEL(03)216-3211㈹
● 東北〈仙台〉(022)264-2131 ● 金沢(0762)63-7621 ● 長野(0262)26-8222 ● 松本(0263)36-7511 ● 静岡(0542)54-9131 ● 名古屋(052)201-8611 ● 京都(075)255-3211 ● 大阪(06)344-1101 ● 広島(082)221-2288 ● 九州〈福岡〉(092)411-6311

資料請求No.45

MICON
変換とバッファの魔術師デブ38X いま登場。
セントロとRS232Cに用途を絞った変換器を作ってみました。セントロのインターフェース規格はパソコンと
プリンタの接続に使います。RS232Cの規格はパソコンとプロッタの接続に、米国製ワークステーシ
ョンとプリンタの接続に使います。
セントロ規格の漢字プリンタを米国製ワークステーションのプリンタに流用する。の例の
場合にセントロとRS232Cの変換器が必要になります。印字出力を別室で重複して
印字したい時にも変換器が必要になります。入力データをセントロ出力と同
時に、RS232Cに変換して別室まで送り、別室でセントロに再変換し
て同型のプリンタで印字させる訳です。
変換器に貯蔵バッファを付加してみました。プリンタに対
する印字待ちが解消できます。プロッタに対する作
画待ちが解消できます。貯蔵バッファはセン
トロとRS232Cで共同利用します。別々
に分割して占有して使うよりバッ
ファの利用効率が向上します。
DEB-381 貯蔵容量 256Kバイト 99,800円
DEB-384 貯蔵容量 1メガバイト 139,800円
印刷カタログのFAX電送＋郵送サービスは0120-03-6001（無料のFAX）でご請求ください。
マイコン工業株式会社 〒150 東京都渋谷区桜丘町14-6
お問い合わせは……マイコン・インフォメーション・センター TEL.03-476-6000(代)
資料請求No.46

夢をかたちに 信頼と創造の｜富士通

こんなプリンタが欲しい

小型で、多機能。そのうえローコストなら申し分ない。

だったらコレ 小型軽量型プリンタ

M3357C/M3358C/M3362C

思い切った小型・軽量化を実現しました。各種機能を凝縮してオプションも多彩。待望のローコスト版です。
●印字方式：24ワイヤドットマトリクス●印字速度（漢字）：M3357C/M3358C＝40字/秒、M3362C＝60字/秒

何よりも使い易さが第一。操作が簡単なのがいい。

だったらコレ 操作性重視型プリンタ

新発売
M3366C/M3367C

オペレータの操作をできるだけ簡単にしました。用紙の吸入、連続用紙の退避など、パネル操作は全てワンタッチ。
●印字方式：24ワイヤドットマトリクス●印字速度（漢字）：M3366C＝48字/秒、M3367C＝64字/秒

印字はとにかく速く。しかも鮮明なのが欲しい。

だったらコレ 高速本格派プリンタ

M3360C/M3361C

24ピンで漢字110字/秒（M3361C）の高速印字。高印字品質で、コストパフォーマンスも向上しました。
●印字方式：24ワイヤドットマトリクス●印字速度（漢字）：M3360C＝80字/秒、M3361C＝110字/秒

超高速なのに印字音が静かなプリンタある？

だったらコレ インパクトラインプリンタ

新発売
M308X シリーズ

ムービングコイル式印字マグネット採用で超高速を実現しました。しかも騒音レベルは55dBA以下。
●印字方式：ワイヤドット式インパクトプリンタ●印字速度（普通漢字モード）：M3081BC/BR＝270行/分、M3082BC/BR＝400行/分
※上記印字速度は、ドット密度（縦横とも）が180dpiの場合です。

●本広告に掲載の全商品ならびにそれに関連する消耗品等および役務について、ご購入の際、消費税が付加されますのでご承知おき願います。

OEMプリンタ

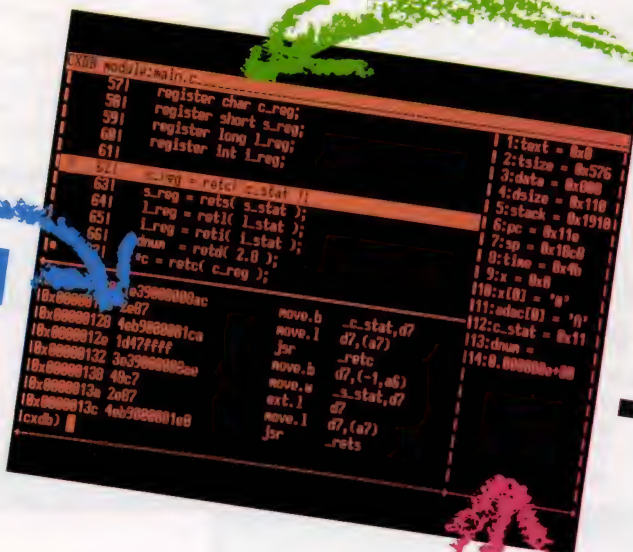
富士通株式会社　国内OEM統括営業部　〒135 東京都江東区東陽4-1-13　☎(03)615-4632～4639（ダイヤルイン）
大阪支店OEM販売部　〒530 大阪市北区堂島1-5-17　☎(06)344-1101㈹

資料請求No.47

CXDB／Cソースレベルシミュレーションデバッガ

はじめに　CXDBはWhitesmiths. Ltd.系のCクロスコンパイラを利用して作成した、プログラムのテストとデバッグを行なうためのCソースレベルシミュレーションデバッガです。
CXDBはより多くの環境で使用できるように、マンマシンインターフェースは扱いやすいウィンドウに設計されており、デバッグ対象プロセッサの実行をホストマシンの環境上でクロスシミュレーションします。
CXDBは対象プロセッサの完全なアドレス空間をアクセスすることができ、周辺装置やハードウェア割り込みのような無作為イベントのシミュレーションも可能です。
CXDBは起動時の指定で、スクロール機能付ウィンドウモードおよびラインモードのいずれかを指定できます。
そして、CXDBは(株)アドバンスド・データ・コントロールズが独自に開発した製品ですので、十分な技術サポートを伴って販売されます。

ソースウィンドウ
■Cのソースコードの表示
■逆アッセンブルの表示
■反転➡現処理行
各種処理を行なうときの対象となる行を指す。
■@➡現実行行
プログラムカウンタが指すアドレスをもとにデバッグ情報から得た行を指す。
■＊➡ブレークポイント行

コマンドウィンドウ
■コマンド入力
■コマンド実行結果の表示
一貫した単純なコマンドを用意しており、これらのコマンドをマクロとして自由に組み合わせ、定義して使用することにより複雑な処理を行うことが可能です。

ワッチウィンドウ
■指定した変数もしくはレジスタの内容を常時表示

■コマンド一覧(その他に21コマンド有り)

A	命令の直接実行	I	機械語レベルのステップ・トレース
a	ラインアセンブラ	i	機械語レベルのステップ実行
B	ブレークポイントの表示	K	マクロの削除
b	ブレークポイントの設定	l	行・アドレス情報の表示
C	Cソース表示モード 注1)	M	マクロ登録表示
c	Cソースリストの表示 注2)	m	マクロ登録
D	全ブレークポイントの解除	me	マクロ登録、修正(ラインエディタ)
d	ブレークポイントの削除	N	次行ステップ実行
E	逆アセンブル表示モード 注1)	n	次行ステップ・トレース
e	逆アセンブルリストの表示 注2)	P	C・逆アセンブルマージモード 注1)
exit	コマンドの中断	p	C・逆アセンブルマージリスト表示 注2)
f	関数情報の表示	quit	CXDB終了
G	トレース	q	CXDB終了(確認付)
g	実行	r	エントリ関数の引数設定および再実行
h	コマンド履歴表示	s	ラインステップ実行

注1)ソースウィンドウへ出力　注2)コマンドウィンドウへ出力

〒170 東京都豊島区北大塚1丁目13番4号　日本生命大塚ビル　TEL.03-576-5351

I·O DATA

プロフェッショナルRAMボード、2タイプ！

最高級スタティックRAMを採用。高信頼性と長期バックアップを同時に実現。RAMボードの理想を実現しました。

OAでの用途はもちろんFAシステムなどで、フロッピィディスクの使用が困難な時に実力を発揮する自動IPL機能。万一の場合に安心な記憶データ保護回路など、バックアップタイプのRAMボードにふさわしい安全設計が随所に。データを大切にするプロの要求に応えます。

■ボード上の電池だけで、プログラムやデータを累計3〜6年間バッテリ・バックアップ。最高級スタティックRAMを採用し、プログラムやデータを累計3〜6年間バックアップ。また、電池電圧のチェック機能を内蔵しているので電池交換も安心です。

■自動IPL(イニシャル・プログラム・ロード)機能を搭載。パワー・オン即自動立ち上げを実現。データ記憶用RAMとは別に、IPL(イニシャル・プログラム・ロード)専用無停電RAM(16KB)を内蔵しています。このため、フロッピィディスクやハードディスクをまったく使用せずに、プログラムの立ち上げが可能。OAはもちろんFA環境下では、必須の機能を確保しています。

■実際の使用を考えつくしたプロ仕様。独立した電源ユニットが不要なプロ仕様です。このため、AC電源そのものが深夜、分電盤側で切断されても安心。オフィスでの高信頼ビジネスユースにぴったりです。

■ハードウェア的なプロテクション機能を確保。二重三重の記憶データ保護設計。

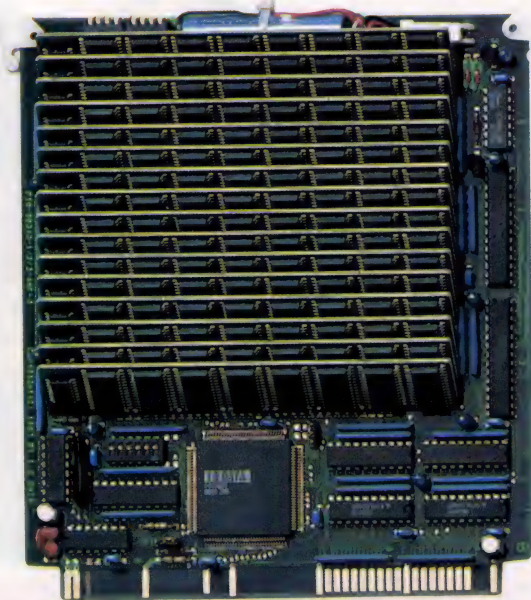
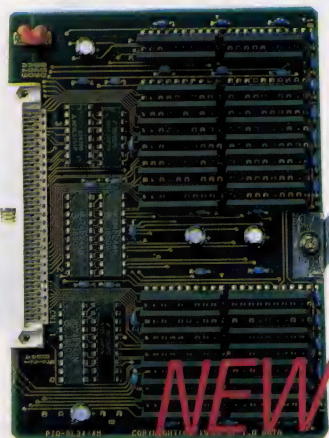
■倍速アクセス機構を搭載。80286CPU搭載パソコン(リアルモード)で、従来比約2倍の高速性を確保。

■確実・超高速なデータチェックを行うHiSUM機能を搭載。

■他のI・Oバンク方式増設RAMボードとの互換性を確保。

スタティック 無停電 増設RAMボード

PIO-9834L Series (1M/2M/3M/4MG)

1MB…¥90,000 2MB…¥165,000 3MB…¥240,000 4MB…¥315,000

■対応機種 PC-9801/E/F/M/U/VF/VM/UV/VX/UX/CV/RA/RX/EX/ES、FC-9801/V/X/A ●リアルモード対応 ●メインメモリ容量128KB、256KB、384KB、512KBの時、増設可能 ●標準メモリ容量640KBのパソコン機種はメインメモリ容量を128KB切り離して増設(I・Oバンク方式増設RAMボードはmax32Mバイトまで増設可能)
■実装方法 標準拡張スロットで行います。(ラップトップ機など、標準拡張スロットを持たない機種では、ご利用できません。)
■対応サポートソフト IOS-10SR RAMディスク(PIO-9834Lシリーズに添付〈5"2HD〉)(アプリケーションを自動立上げするソフト)

80386ワールドを、より高度なプロフェッショナル環境へ――。最上位機種PC-98RL対応高速32ビットRAMボード。

次代のプロフェッショナル環境を創造する高速32ビットRAMボードです。最高級機種活用のプロフェッショナルニーズに応える充実度を誇っています。

■80386CPUの高性能をフルに引き出す32ビットデータバス対応。特に、マルチタスクでは抜群の威力を発揮します。32ビット80386CPUの高性能をフルに引き出し、技術計算やAI分野などの高度なニーズに応えます。また、複数のアプリケーションを平行して実行する本格的なマルチタスク環境にも余裕をもって対応。特に、次世代OS/2環境下では必須の高速32ビットRAMボードです。

■NEC純正品PC-98RL-01＋PC-9801-54×2とフルコンパチブルで、メモリ専用スロットに実装可能です。

NEC純正品PC-98RL-01＋PC-9801-54×2と完全互換性を保ち、コストパフォーマンスも抜群。PC-98RLのメモリ専用スロットに実装するため、貴重な汎用拡張スロットを使用しません。

■対応機種 PC-98RL2/RL5 ■実装方法 メモリ専用スロット(32ビットデータバス)に実装 ■対応サポートソフト IOS-10/386(アイ・オー・データ)、MEMORY-PRO386(メガソフト社)、WINDOWS/386、OS/2(NEC)

RL専用4MB 実装済 増設RAMボード

PIO-RL34-4M

4MB(実装済)…¥148,000

I·O DATA パソコン増設RAMのトップブランド 株式会社アイ・オー・データ機器
本社／サポートセンター 〒920 金沢市駅西本町1-5-41 TEL.0762-21-4812 FAX.0762-24-9300
東京営業所 〒101 東京都千代田区神田富山町6 松崎ビル4F TEL.03-254-0301 FAX.03-254-9609
※広告掲載の表示価格には消費税は含まれておりません。

パソコン組み込み型

ニューロ・コンピュータ

4DSPのリング結合並列アーキテクチャを採用し、平均10M CPSで高速処理

■特　長

●パソコンの拡張スロットに組込み、ニューラル・ネットの高速演算及び開発が可能 ●演算素子として、24ビット浮動小数点DSP(富士通MB86220)を4個使用、リング結合並列アーキテクチャ ●平均10M CPS(CONNECTIONS/SECOND)の高速処理 ●データ・メモリ容量798KB～3.1MB SRAM ●DSPのプログラムメモリは8KWの高速SRAMこれにより、ニューラル・ネットの各種応用に柔軟に対応可能 ●低価格

■バックプロパゲーション・ソフト仕様

●ネットワーク構造/3層構造ネットワーク ●ネットワークの規模/最大ニューロン数：各層1,000個 最大結合数：MINシステム最大100K個 MAXシステム最大400K個 ●処理速度/2.0M CPS(学習時平均) ●学習機能/学習係数の変更、学習回数及び最大トータル誤差の設定、トータル誤差の表示(リアルタイムでの表示)、学習時間の表示 ●ネットワークのグラフィック表示/結合係数、ニューロンの出力 ●結合係数のセーブ・ロード(バイナリファイル)/学習中及び学習を終了したネットワークの結合係数をファイルに保存できる。初期値の結合係数のロードも可能 ●学習及び認識データファイル/テキストエディタなどにより簡単に作成可能。ファイルにてデータ渡し ●コマンド入力/ウィンドー機能によって簡単に操作可能 ●稼動環境/MS-DOSバージョン3.1以上

■価　格

製品	価格
●NEURO・TURBO－MINシステム(バックプロパゲーション・ソフト付)	980,000円
●NEURO・TURBO－MAXシステム(バックプロパゲーション・ソフト付)	1,480,000円
●NEURO・TURBO－MINボード	880,000円
●NEURO・TURBO－MAXボード	1,380,000円
●バックプロパゲーション・ソフト	100,000円

※開発中ソフト
●高速バックプロパゲーションソフト
●連立多元方程式の高速解法ソフト
●その他のニューラル・ネット・ソフト

NEURO TURBO

ニューロ・ターボ

パソコンPC-9801に組み込み

※記載されている価格には消費税は含まれておりません。

本製品は名古屋工業大学と当社ニューロ・コンピューティング研究所にて共同開発致しました。

株式会社マイテック
〒171 東京都豊島区高田3-32-1 大東ビル5F
TEL.03－987－7400　FAX.03－983－5505

資料請求No.51

MINATO

なんと¥298,000(ニッキュッパ)で、高速コピー専用器登場。

NEW

ギャングプログラマ MODEL 1891

¥298,000

メガビット時代のコンパクトコピー専用ギャングプログラマ<MODEL 1891>は、薄型・軽量のボディにギャングソケットを搭載、標準で32ピン4Mビットまでのプロム8個にデータを同時にプログラミングできます。バッファメモリを装備しているのでマスターROMを選びません。視認性の優れた32文字ディスプレイは、PROMの型名やVpp電圧、書き込みパルス幅などはもちろん各種の設定項目をメニュー型式で表示するのでオペレーションは、快適です。

●16k～4MビットEPROM、EEPROM(32ピンまで)●シリコンシグネチュア機能●高速書き込み機能搭載●マスターROMを選ばないメモリ方式●200種以上のPROMをサポート●多彩な信頼性チェック機能●デバイスメーカ各社の書込みアルゴリズムを装備●4Mビットバッファメモリ装備

書き込み時間データ(単位：sec.)

デバイス名	ブランクチェック	プログラム	ベリファイ	連続モード
P27512 (512Kビット)	1.5 sec.	14.5 sec.	5.5 sec.	21.5 sec.
i27270 (1Mビット)	3 sec.	22 sec.	10 sec.	35 sec.
μPD27C2001 (2Mビット)	5 sec.	43 sec.	20 sec.	68 sec.
μPD27C4001 (4Mビット)	10 sec.	87 sec.	39 sec.	135 sec.
TC574000 (4Mビット)	10 sec.	108 sec.	39 sec.	156 sec.

※40ピンメガビットPROM対応ギャングソケット<TYPE9140A>近日発売

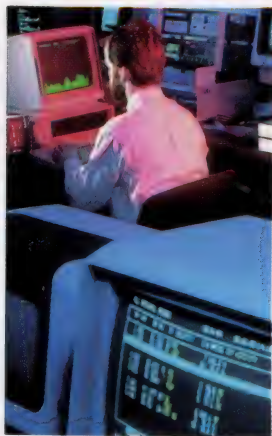
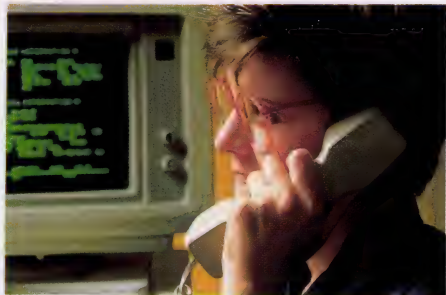
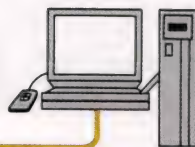
ミナトエレクトロニクス株式会社

本　　社／223横浜市港北区南山田町4105　☎045(591)5611㈹
大阪営業所／530大阪市北区曽根崎新地2-6-21　☎06(348)9277㈹
北関東営業所／370高崎市双葉町6-25　☎0273(23)9701㈹
福岡営業所／812福岡市博多区博多駅東1-9-5　☎092(475)2825㈹
仙台営業所／980仙台市宮城野区榴岡2-2-10　☎022(291)2571㈹

資料請求No.52

UNIX、咲かせましょ〜う。
vi
awk
cat
NFS
ksh-i
#
B-NET
ls
$
ar
%
cc
MNLS
信頼に応えるNUCの移植サービス。
日本ユニソフト株式会社(NUC)は、移植のエキスパート企業として、OS、言語プロセッサ、アプリケーションの提供を行っています。
特にUNIXに関しては、米国UniSoft社(カリフォルニア州エミリビル)と技術提携し、先進の移植技術、環境を導入。UNIX OSのカスタム化サービス、周辺ソフトウェア、クロス環境開発ツールの提供、さらに日本語マニュアル製作まで、実効性の高いトータルなUNIX稼働環境を創出します。
また、NUCは、米国UniSoft社が提供するUniPlus⁺を日本国内に供給。UniPlus⁺は、AT&T標準リリースのUNIX System Vを機能強化した豊富な拡張機能を使用でき、全世界で100以上のコンピュータ・システムに移植実績があります。
NUCは、UNIXを中心にベーシック・ソフトウェアの移植・開発を通して、トータル・ソリューションを提供していきたいと考えています。ソフトウェア開発環境でお悩みの点は、当社営業部にお気軽にご相談ください。
●UNIX国際版移植
●UniPlus⁺(UNIX拡張版)移植
●言語、日本語環境
●ネットワーク環境開発
Trademarks
UniPlus⁺: UniSoft Corporation.
UNIX OS was developed by Bell Laboratories and is licenced by AT&T.
NUC 日本ユニソフト株式会社
〒102 東京都千代田区三番町8-7 第25興和ビル ☎(03)237-3321

資料請求No.54

この環境を選ぶ、と。また、一歩差をつける。

LAN NEWS

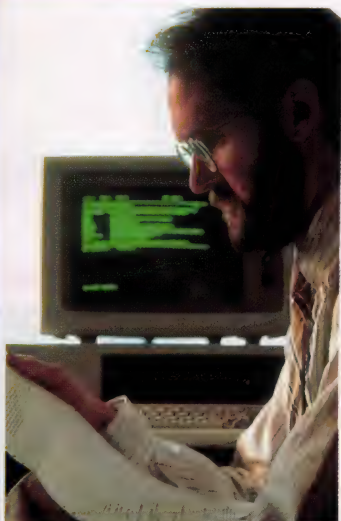
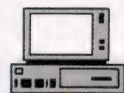
ソフトウェアを９月にバージョンアップします。
EXOS 8098 V3.2.3→V3.5

主な内容：

●FTP転送スピードアップ1.5～3倍

●Rユーティリティの追加（rsh, rcp, rexec, rprなど）

●ソケットライブラリでLattice Cサポートを追加

IEEE 802.3
10 BASE 5

お手持ちのPC9801やFMRなどのPCと、SUNやNEWSなどのEWSをTCP/IPで結ぶEXCELAN社のLANシステム。パソコンの世界が見違えるほど拡がり、EWSに標準のEthernetを存分に活かせます。

X Server/DOS

PC9801上でXwindowを実現します。PCで、Xwindow環境が使える画期的なソフトです。

PC-NFS

「ディスク容量を増やしたい。」「レーザプリンタを使いたい。」共有すればPCにディスクを増設した感覚で使えます。

SMB Server

MS-NetworksなどにEWSを接続しファイルサーバ、プリンタサーバを実現します。

JMCネットワーク・フェア'89のご案内
LANに関する米国の最新技術動向、アプリケーションソフトの実際など、米国EXCELAN社、SYNTAX社の技術者によるセミナーを行います。
●日時：9月20・21日のいずれかをご指定下さい。・セミナー1:00～5:00・パーティー6:00～8:00●場所：外苑前TEPIA●お問合せ　044-711-0330担当：松田お待ちしております。

EXOS 10698	EXOS 10691	EXOS 10691	EXOS 205	EXOS 215	EXOS 10624	EXOS 10642	EXOS 10633	EXOS 10634
PC-9801	FM-R	PanacomM	IBM PC/AT	IBM PS/2	VAX	MicroVAX	Tower 16/32	Intel 310
MS-DOS	MS-DOS	MS-DOS	PC-DOS	PC-DOS	VMS	MicroVMS	SystemV	XENIX
			XENIX, SystemV		4.3BSD, SystemV	4.3BSD		

PC9801は日本電気、FM-Rは富士通、PanacomMは松下電器産業、IBM PC/AT、IBM PS/2、PC-DOSはIBM、VAX、MicroVAX、VMS、MicroVMSはDEC、Tower 16/32はNCR、Intel 310はIntel、MS-DOS、XENIXはMicroSoft、Unix、SystemVはATT、SUN、NFSはSUN Microsystem、NEWSはSONYの商標です。

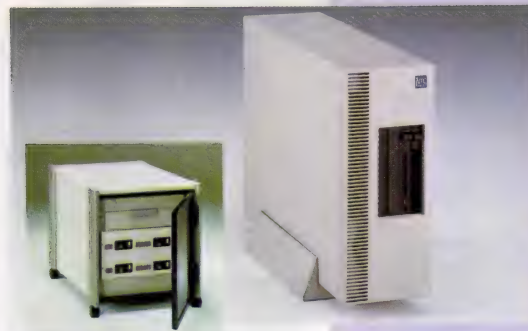
お問合せは　本社 044-711-0330　大阪 06-300-0810

JMC ジャパンマクニクス株式会社

本　社／〒211 川崎市中原区今井南町516
大阪営業所／〒532 大阪市淀川区西中島5-11-8（新大阪木村第一ビル）

資料請求No.55

More performance for your SUN...

超高速!512KBキャッシュメモリー搭載!大容量ハードディスク!

ワークステーションにおいて2次記憶装置のスピードは、システム全体のスピードに大きな影響を与えます。転送速度の遅いコントローラーを用いると、システムのトータルパフォーマンスが著しく低下します。NTHシリーズは平均アクセスタイム16ms以下のハードディスクと、512KBキャッシュメモリー搭載のシプリコ社SMD-Eコントローラーボード(RIMFIRE3200シリーズ)の採用でシステムのポテンシャルを最大限まで引き出すことが可能です。

増設ハードディスクドライブサブシステム for Sun-3/4

【SMD-Eタイプの適応機種】
●Sun-3:75/110/140/150/160/260
●Sun-4:260

●SMD-E対応のシプリコ社コントローラーボードRF-3220採用。●NTHハーフワイドケース(幅385×奥行900×高さ730mm)SUN純正ペデスタルボックスと同じ高さです。●他に19インチタイプケースも供給可能です。(幅600×奥行900×高さ680mm)

VME対応NTHシリーズ

●大容量560MBから1.6GBまで。(フォーマット後容量)●適応機種SUN-3(110/140/160/260)、SUN-4(260)●NTH-560E-8(1スピンドル、容量560MB)●NTH-800E-8(1スピンドル、容量800MB)●NTHH-1.1G-8(2スピンドル、容量1.1GB)●NTHH-1.6G-8(2スピンドル、容量1.6GB)

ESDI対応NTHシリーズ(SCSIポートに接続)

●容量330MBから660MB(フォーマット後容量)、オプション対応として¼インチカートリッジストリーマー搭載可能。ケースはデスクトップ型です。●外形寸法:幅155×奥行550×高さ380mm ●適応機種:SUN-3(50/60/75/110/140/150)、SUN-4(110)●NTH-330E-5A(1スピンドル、容量330MB)●NTH-660E-5A(1スピンドル、容量660MB) ●NTH-330E-5MT(1スピンドル、容量330MB、¼インチCMT付属)●NTH-660E-5MT(1スピンドル、容量660MB、¼インチCMT付属)●一筐体の中に5インチディスクを2台まで収納できます。●仕様はSUN純正品と全く同等です。●電源、コントローラーボード、2メートルSCSIケーブル(シールドタイプ)等付属品を含みます。

*上記標準品の他、お客様のご要望により調整可能です。詳しくは弊社担当者までお問合せください。*ハードディスクサブシステムは、設置費用、インストレーション費用が別途となります。*尚、上記製品は東芝ASシリーズにも接続可能です。

▶SUN-3/60用4MB増設メモリーキット……定価¥320,000
▶SUN-4/110用16MB増設メモリーキット……定価¥960,000

PS3110 advanced memory expansion plus SCSI port interface for SUN-3/100 Series.

●Capacity to 24Mbytes Per Card ●Low-Power 1Mb DRAM Technology ●Provision For On-Card SCSI Host Adapter ●Brings Pedestal Functionality to Desktop Systems

PS3100シリーズ SUNワークステーション用増設メモリーボード

■特徴:当製品はメモリーボード上にダブルハイトシングルデプス標準VMEボードを搭載できますので、特にSun-3/75, 3/110, 3/140, 3/150のように拡張スロットルの少ない機種に最適です。

型 番	容 量	サイズ
PS3110-04	4メガバイト	Width:366.7mm. (14.44") Height:400.0mm. (15.75")
PS3110-08	8メガバイト	
PS3110-12	12メガバイト	
PS3110-16	16メガバイト	
PS3110-20	20メガバイト	
PS3110-24	24メガバイト	

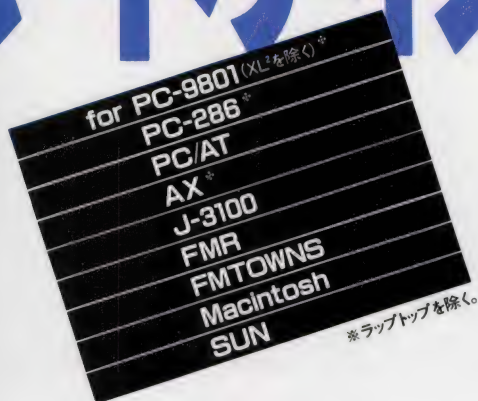
■接続可能機種:
Sun-3/75, 3/110
3/140, 3/150
3/160, 3/180

*一般に、各商品名は各社の登録商標または商標です。

★商品価格には消費税は含まれておりません。

■日本総代理店:CIPRICO社/PARITY SYSTEM社/TECH-SOURCE社

NEWTECH CO.,LTD.
株式会社ニューテック　担当:荒田/風越
〒113 東京都文京区湯島1-3-6 Uビル7F
TEL:03-813-3891(代) FAX:03-813-3894

145MB 245,000円 ハードディスク新発売

ハードディスクの価格と機能の常識をCTSが大きく変えました。わが国で初めて、コンポーネントタイプのハードディスク《HDα》の登場です。《HDα》は、その低価格性もさることながら、ハードディスク本体を電源ユニットとディスクユニットと拡張ユニットの3つのユニットにコンポーネント化。ですから、将来、メモリの増設が必要になったときにはディスクユニットの取り替えコストで拡張できるわけです。同じように、拡張ユニットは、将来、テープストリーマやフロッピーディスクの増設ユニットとしての活用が可能です。つまり、ハードディスク、テープストリーマ、フロッピーが共存する、まったく新しいマルチメディアディスクとして活用できるわけです。

拡張ユニット
テープストリーマ
増設ユニット
近日発売開始！
詳細はお問合せ下さい。

《HDα》PC-9801の主な特長(その他の機種についてはお問合せ下さい)

- 平均アクセスタイム：14ms～23ms
- SCSIインターフェイス採用(SCSIインターフェイスボード標準装備)

《HDα》PC-9801価格(その他の機種についてはお問合せ下さい)

《HDα》-83	198,000円	《HDα》-337	580,000円
《HDα》-145	245,000円	《HDα》-668	980,000円

※各製品は各社の登録商標です。

株式会社 シー・ティー・エス
本　社 〒110 東京都台東区台東2-32-9 第一小林ビル2F TEL.03(835)1887 FAX.03(835)2604

株式会社 シー・ティー・エス販売
本　社 〒110 東京都台東区台東2-32-9 第一小林ビル5F TEL.03(837)4355 FAX.03(835)2604
大阪支社 〒534 大阪市都島区東野田町1-21-20 黒崎ビル3F TEL.06(353)0437 FAX.06(353)0438

資料請求No.58

BITRAN

進化のプロセスは、語れない。

Progress to 80386/V33

時代のCPU動向に対応して、ICEが進化していく。
いま、"80386"そして"V33"環境へ。
BITX-5000シリーズ、ネクスト・ステップへ。
進化のプロセスは、語れない。
時代を代表するひとつの表情が、ビットランから……。

BITX-5000 SERIES
インサーキット・エミュレータ

EDIBUGキャンペーン実施中!!
●期間中EDIBUGが30%OFF
●体験ディスケットを用意しています。
詳しくは☎03(980)5851まで

●CPUの最高速度を完全リアルタイム実行。
80386：20MHz/25MHz, V33：16MHz

●最新の高級言語もソースレベル対応。
MS-C, Lattice-C, Turbo-C,……C386, PL/M386 がソースレベルでデバッグできる。

●エディタとICEが同居 ―― EDIBUG
ソースプログラム作成時に使用しているエディタで、ソースのアクセスができる新感覚のヒューマンインターフェース。

●強力なディバッグ機能。
タスクに影響を与えない仮想ブレークモード、指定タスク内のみでのトレース機能を用意。

●80386の全てのモードに対応。
リアルモードを標準、プロテクトモード、ページングモード、バーチャル86モードをオプション・サポートすることにより、ハイコストパフォーマンスを実現。

●BITXシリーズ対応CPU：8086/8088/V20/V30、80186/80188、V40/V50、V25/V35、80286、80386、V33
●対応ホスト：PC9801シリーズ、PC286シリーズ、J-3100シリーズ、IBM-PCファミリー、IBM-5500ファミリー、AXパソコン等

究極のデバッグを追求する
ビットラン株式会社
東京営業所／東京都豊島区池袋2-53-10　第7日新ビル9F　TEL.03-980-5851(代)　FAX.03-980-2370　〒171
本　　社／埼玉県行田市持田2748　TEL.0485-54-7471(代)　FAX.0485-53-3388　〒361

MEGASOFT

素速く、漏らさずデータ転送。

高速リモートディスクシステム

MAXLINK

マックスリンク

MAXLINK(マックスリンク)は、 2台のパソコンの間でもっとも簡単・高速にファイルやデータの交換を行うことのできる高速データ転送ツールです。

MAXLINKで広がる世界

❶ラップトップとデスクトップとのデータ交換に

3.5インチディスクのラップトップパソコンと5インチディスクのデスクトップパソコンの間で、データの交換に困っていませんか? MAXLINKがあればメディアの違いを越えて自由なデータ交換を簡単に実現できます。

❷旧システムから新システムへの移行作業に

パソコンを新システムに置き換える際に、これまでに蓄積されたハードディスク上のデータやプログラムを新システムに簡単に移行できます。20MBのディスクをまるごと転送してもわずか数十分。昼休みの間にもすっかり移行は完了です。

❸ハードディスク上のデータ共有

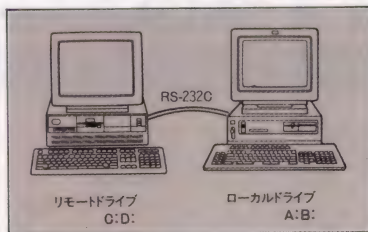
片方のパソコンにハードディスクがない場合でも、MAXLINKで接続したパソコンのハードディスク・RAMディスク・フロッピーディスクなどに自由にアクセス。ちょっとしたLAN感覚でデータの共有が実現します。

❹ハードディスクのバックアップに

MAXLINKを利用してハードディスクの内容を別のバックアップ用パソコンに転送することができます。タイムスタンプをチェックして、更新のあったファイルだけを選択転送することが可能です。バックアップを毎日の習慣にしてしまえば、わずか数分の手間で貴重なデータの安全性が守れます。

❺異機種間のデータ交換に

MAKLINKは同一メーカーのパソコン間だけでなく、他の機種との接続も可能になるよう設計されています。たとえば、PS/55用のMAXLINKとPC-9800シリーズ用のMAXLINKを組み合わせば、それぞれの間での自由なデータ交換も可能になります。

<MAXLINKの特長>

●RS-232Cケーブルだけで高速データ転送を実現 ●PS/55やAT互換機用のパラレル転送モード(PS/55用) ●実効転送速度30,000〜120,000bpsの高速転送(特許出願中) ●異なるメディア間でのデータ交換を実現 ●CRCチェック方式による高い信頼性 ●異機種間でも双方向のデータ転送が可能 ●簡単便利な対話型ファイル操作ユーティリティを装備 ●ディレクトリ単位のファイル転送、日付による選択転送機能 ●リモートディスクドライバで市販ソフトからでも使用可能

●PC-9801シリーズおよびその互換機用　価格18,000円

対応機種/NEC PC-9800シリーズおよびその互換機
製品の構成/システムフロッピーディスク……2枚
(5インチ2DDおよび3.5インチ2DD 各1枚)
(PC-9801/E/Mでは、2DDの読めるディスクドライブが必要です)
専用RS-232Cインターフェースケーブル……1本
マニュアル、ユーザー登録カード……各1部

●PC/ATおよびその互換機用　価格25,000円

対応機種/PC-ATおよびその互換機、AX、J-3100シリーズなど
(AXおよびJ-3100の日本語モードでも使用できます)
製品の構成/システムフロッピーディスク……2枚
(5インチ2Dおよび3.5インチ2DD各1枚)
特別仕様RS-232Cインターフェースケーブル……1本
極性変換用アダプタ……1個
マニュアル、ユーザー登録カード……各1部

●IBM PS/55用　価格30,000円

対応機種/IBM PS/55およびPS/2
製品の構成/システムフロッピーディスク……2枚
(5インチ2DDおよび3.5インチ2DD各1枚)
特別仕様RS-232Cインターフェースケーブル……1本
極性変換用アダプタ……2個
パラレル接続用コネクタ……1個
マニュアル、ユーザー登録カード……各1部

●表示価格には消費税は含まれておりません。

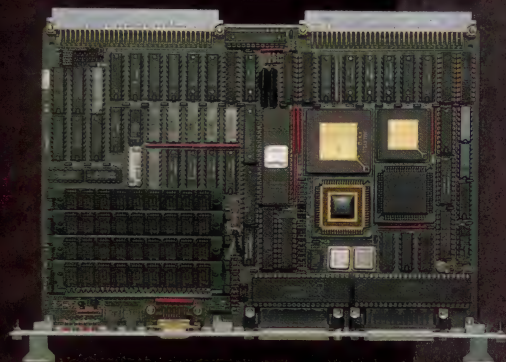
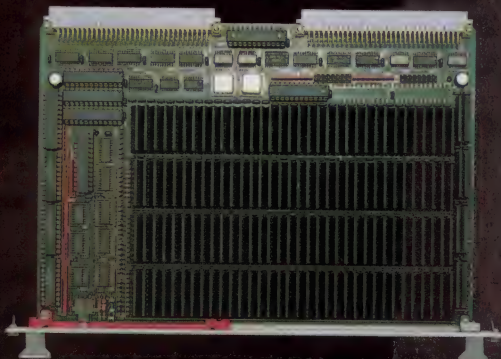
INTER-MEDIAコンセプト
メガソフト株式会社 〒564 大阪府吹田市江の木町16-9 近藤ビル6F
TEL. 06-386-2058(代)　FAX. 06-386-2123

資料請求No.60

ampère

新製品2機種登場です。

アンペールのVMEモジュールMACRO6000シリーズに

MACRO 6020
32ビットMPUモジュール

MACRO 6120
16MBダイナミックRAMモジュール

「MACRO 6020」は、MC 68020を採用した32ビットMPUモジュールで、C-MOSタイプのCPU周辺コントロールICとVMEバス・コントロールICを使用することにより、信頼性の向上を図っています。〈特長〉●MC68020、68000CPU周辺/VMEバス・コントロールICなどを採用した、高性能かつ高信頼性の32ビットMPUモジュール。●RAMのパリティ・チェック機能を有している。●アービタはモードを切り換えることにより、ラウンドロビン選択とプライオリティ方式とを使い分けることができる。●ローカル・バスには、I/Oチャンネルを採用。●メモリ、I/Oはモトローラ社のMVME133/133Aとアッパーコンパチブルで、デバッグソフトやVERSA DOS等のソフトウェアをそのまま使用できる。

「MACRO 6120」は、大容量データの高速処理用に設計された、16MBのダイナミックRAMモジュールで、VMEバスのRev. Cに準拠しています。〈特長〉●メモリ容量は16MBで、アクセスタイムは220ns以内。●32/16/8ビットデータ転送、非整列データ転送、RMW等のデータ転送をサポート。

OEMおよびカスタムボードの設計・製造もご相談ください。

MACRO 6000シリーズ製品一覧

〈VMEモジュール〉
- MACRO 6020 32ビットMPUモジュール
- MACRO 6120 16MBダイナミックRAMモジュール
- MACRO 6002 低消費電力形MPUモジュール
- MACRO 6401 低消費電力形ディスク・コントローラ・モジュール
- MACRO 6302 低消費電力形ROM/RAMモジュール
- MACRO 6001 MPUモジュール
- MACRO 6101B 2MBダイナミックRAMモジュール
- MACRO 6641 32ビット・デジタル入出力モジュール
- MACRO 6551 ユニバーサル I/Oモジュール

〈I/Oチャンネル・モジュール〉
- MACRO 6702B フロッピーディスク・コントローラ・モジュール
- MACRO 6705 ハードディスク・コントローラ・モジュール
- MACRO 6720 デュアルRS-232Cモジュール
- MACRO 6740 デュアル・パラレルポート・モジュール
- MACRO 6741 GP-IB I/Fモジュール
- MACRO 6743 デュアル・パルスモータ・コントローラ・モジュール
- MACRO 6744 16ビット・デジタル入力モジュール
- MACRO 6745 16ビット・デジタル出力モジュール
- MACRO 6753 デュアル・パルスモータ・ドライバ・モジュール
- MACRO 6780 16チャンネル・アナログ入力モジュール
- MACRO 6781 4チャンネル・アナログ出力モジュール
- MACRO 6501 カードラック
- MACRO 6541/6542 ユニバーサル・ボード
- MON68K, MON68K/20 デバッグ・モニタ
- Hiterm インテリジェント・ターミナルエミュレータ
- MACROS 68000 マイクロ・コンピュータ・システム
- AML66Kシリーズ 68000CPU周辺コントロールIC VMEバス・コントロールIC

製造元	株式会社　アンペール　〒160 東京都新宿区西新宿7丁目5番20号 TEL 03(365)0825(代)				
販売代理店	㈱ロッキー 東京 03-228-4511 大阪 06-300-1395	日本電子機材㈱ 東京 03-253-3188 名古屋 052-762-1355 大阪 06-385-6707 京都 075-311-3071 福岡 092-573-4572	シルウォーカー㈱ 東京 03-341-3651 大阪 06-303-4102	日本アイ・ディ・アイ㈱ 川崎 044-922-0711	株式會社 S.Y. Engineering Systems 서울 02-593-3301〜2

資料請求No.61

書換え可能型650MB光磁気ディスク装置

NEW HP-IB

MOD5-HP650

MOD5-HP650光磁気ディスクは書換え可能であるため、従来の磁気ディスクパック、磁気テープなどと容易に置き換えられ、かつ信頼性、操作性、経済性を飛躍的に向上させることが可能です。

¥1,896,000

HP互換
* HP-IBとCS/80プロトコルのフルサポートによりHP1000/3000/9000全てのコンピュータでの使用が可能。
* HP標準形状に合致した筐体デザインによりHPのミニラックに容易に収納可能。

大容量
* 扱い易い5.25インチカートリッジ一枚で約650MBという、HP7907A8インチカートリッジの30倍の容量のデータが記録可能。
* オートチェンジャの使用により数十～数百GBのオンラインストレージも実現可能。

高速性
* 高速回転ドライブ、高速インターフェイス技術により光ディスクとしては最高のパフォーマンスを達成

スペースファクタ
* オープンリールに比べてメガバイト当り1/30、IBM3480型カートリッジテープに比べてメガバイネ当り1/4。

高信頼性
* ビットエラーレート10^{-12}の高い信頼性を実現。
* ISAのユーザ本位の設計ポリシーと高度な安全設計技術は製品の企画から生産工程まで明確にその主張を貫いています。
* ディスクモータの誘動負荷を充分考慮した高性能ディスク専用電源の採用。
* 信頼性・パフォーマンスを第一に評価・採用したデバイス。

2.3GBインテリジェント・テープバックアップ装置

NEW HP-IB

ET8200-HP

HP1000／3000／9000フルサポート

¥1,968,000

低ランニングコスト
* 8ミリビデオテープ使用による低ランニングコスト。

大容量
* 最大2.3GBの大容量。

高信頼性
* 強力なエラー自動訂正機能により、10^{-13}という極めて低いエラーレートを達成。

高速性
* 10MB／分という高速バックアップが可能でHPのカートリッジテープ装置に比べ約5倍のスピードを確保。

HP互換(オンラインモード)
* HP標準筐体形状に合致。
* HP79XXタイプのオープンリールテープ装置をエミュレート。
* 各種OSにおける多くのシステム・バックアップ・コマンドをサポート。

多機能性・操作性(オフラインモード)
* CS80コマンドセットをエミュレート。
* テープ装置自身がコントローラとして各デバイスを制御。
* マルチディスク・サポート。
* スケジューリング・サポート。
* インタラクティブ・パネル機能。
* ステータス情報のLCDへの表示。
* すべての設定はホストからもプログラミング可能。
* リアルタイム・クロック内蔵。
* バッテリーバックアップによるプログラミングデータの保存。

お求めは全国の当社代理店または当社国内営業部03(232)8570まで

WE ARE HP PROFESSIONALS

ISA CO., LTD.

株式会社 アイ エス エイ

〒169 東京都新宿区大久保2-6-16平安ビル

資料請求No.62

【ICE+ROMエミュレーター=iD-1600】

デバッグ環境の革新見誕生!

高性能ユニバーサルインサーキット・デバッガ

iD-1600

次々と発表されるニューCPU。性能面でも高速化の一途をたどっています。iD-1600は、これらCPUへの柔軟な対応と、すぐれたコストパフォーマンスの追求をテーマに開発された新世代のデバッグツール。P-ROMソケットに専用プローブをインサーキットするだけで、ターゲットCPUのリアルタイムデバッグを可能にしました。つまり、ROMエミュレータ感覚の実に手軽なシステム構成で、ICEに匹敵する高度な機能を備えたデバッグツールが手にできるのです。しかも、この方式ですと、同一ハードウェアで、高速化・高機能化するニューCPUの数々にもマルチに対応できます。
「これからは、iD-1600だね」という方が増えています。この紙面を読み終えたとき、あなたもその一人になっていただけるはずです。

NEWS

iD-1600(68K):68000対応モデル '89.4Q発売予定
PLB-200 :iD-1600と組み合わせて利用可能な200MHz対応ロジックアナライザ(単体使用も可能) '89.4Q発売予定

■iD-1600(86対応モデル)

対応CPU	8086/88 80186/188 80286(リアルタイムモード) V20/V30 V40/V50 V25/V35
対応Cコンパイラ	MS-C Lattice C Turbo-C (ROM化のためのスタートアップルーチン付)
Cソースレベルデバッガ	Cdeb 86付

■iD-1600(80対応モデル)

対応CPU	Z80 64180 TMPZ84シリーズ 8085
対応Cコンパイラ	LSI-C80(MS-DOS版) HITEC-C80 (ROM化のためのスタートアップルーチン付)
Cソースレベルデバッガ	Cdeb 80付

■対応P-ROM

対応機種	2764 27128 27256 27512
アクセススピード	150ns

尚、オプションで1Mビット対応プローブがあります。但し、モデル2でのみ使用可

■ホスト環境

NEC PC-9801シリーズ(XA/XL/RLハイレゾも可)
IBM-PC/AT
尚、NEC PC-9801シリーズとIBM-PC/ATシリーズでは専用I/Fカードが異なります。ご注文時にご指定ください。

■その他の特長

- マクロ言語、マルチジョブ、テンプレート/ヒストリー、ロギング/パッチの各機能
- マルチウインドウ対応
- プログラムブレーク(MAX15点)、強制ブレーク、ステップ実行、ターゲットCPU動作モニター リアルタイム・カウントの各機能
- ハードウェアブレーク、リアルタイムトレース機能(モデル2のみ可)
- ホストのパソコンとは専用パラレルI/Fで接続し高速処理を実現
- ROMエミュレータ機能もサポート可

■価格

モデル1	iD-1600(86) iD-1600(80)	各¥198,000
モデル2	iD-1600(86) iD-1600(80)	各¥298,000
オプションソフト	C deb 86 C deb 80	各¥98,000
オプションプローブ	1Mビット対応用(モデル2のみ使用可)	近日発売

モデル1:エミュレーションROM容量128KB
モデル2:エミュレーションROM容量256KB、リアルタイムトレース機能/ハードウェア・ブレーク機能をサポート

※PC-9801シリーズは日本電気社、MS-DOS、MS-Cはマイクロソフト社、Lattice CはLattice社、Turbo CはBORLAND社、LSI-C80はエル・エス・アイジャパン社、HITEC-C80はハイテックソフトウェア社の登録商標です。

COMPUTEX
コンピューテックスジャパン株式会社
〒101 東京都千代田区神田和泉町1-3-7 西川パーキングビル
TEL:03(5687)1201(代) FAX:03(5687)1202/関西地区お問い合わせ先TEL075(551)0373

資料請求No.63

TOKEN PASSING BUS LAN

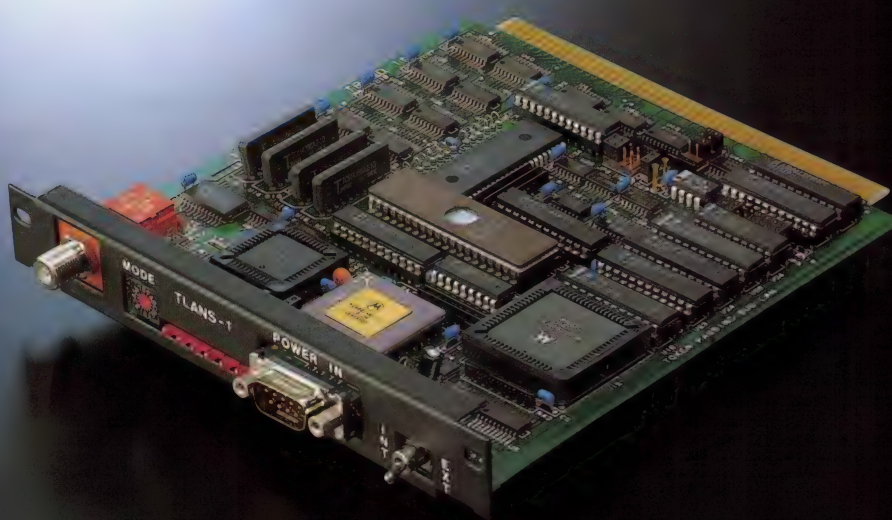
TLANS-1

TLANS-1

MODE

POWER IN

INT

EXT

OSIレイヤー2	IEEE 802.2 LLCタイプ3
	IEEE 802.2 MAC
OSIレイヤー1	IEEE 802.4 Phase Coherent FSK

TLANS-1(TREND LOCAL AREA NETWORK SYSTEM)は、MS-DOSの内部コマンド (DIR, COPY etc)を使用し、ステーション間のファイル転送や、他のディレクトリーの表示、変更、削除などの操作が、誰にでも簡単にできます。また、数kByteの小さなデバイス・ドライバーをサポートしている為、既にワープロ等で使用しているプログラムはもちろん、新しく作るプログラムも何の特別な方法をとらずに、LANを構築することができます。※順次、ミニMAP、MMSサポート予定。

【ハードの主な仕様】
- CPU/MC68000(12.5MHz) ● ROM/128KB、RAM/512KB
- TBC/68824、TBM/82511 ● 伝送速度/5Mbps(10Mbps切換可能)
  ミニMAP適合のLANインターフェイスを提供RAMはPCバス側からアクセスできる共有RAMです。

【ネットワークの主な機能】
- ファイルの共有/ネットワーク内の複数のディスクを対等にアクセスできます。
- 周辺機器の共有/ネットワーク内の複数の周辺機器を対等にアクセスできます。
- データ・プログラムの共有/自端末以外のディスク、周辺機器を自端末に結合されているように使用できます。

TLANS-1 ¥198,000

●MS-DOSは、マイクロソフト社の登録商標です。※表示価格には消費税は含まれておりません。

トレンドは、TLANS-1を使用しFAにおける多種多様のシステム導入のためのプランニング、コンサルティングを全てサポートし、お客様のニーズを満たすシステム作りを致します。

【お問い合わせ】
〒651 神戸市中央区磯上通6-1-11 三井生命神戸ビル
サポート専用電話 TEL(078)231-4853
FAX.(078)231-4846

資料請求No.64

27のユーティリティを統合。先進の機能がきらめく。

日本語版ノートン・ユーティリティーズ アドバンスト エディション

新発売

ノートン・ユーティリティーズ・アドバンスト・エディションは、MS-DOSとハードウェアをフル活用する27の機能を統合した日本初のユーティリティ。ディスクアクセスを高速化する「スピードディスク」やディスクのエラーを自動診断・修復する「ノートンディスクドクター」。また、消去ファイルを素早く簡単操作で修復できる「クイックアンイレーズ」、ミスフォーマットしたディスクを復旧する「フォーマットリカバー」など多彩で広範な機能を結集。比類のないパワーを発揮します。27機能の起動は、メニュー画面、コマンドラインの両方からOK。ユーザレベルを問わずに高度な機能が駆使できる、初心者にも安心の使い易さ。いま、ピーターノートンが、あなたのパソコン環境を一変します。

スピード・ディスク

ディスクの最適化を行なう最新・最強のユーティリティ。ディスク内の断片化したファイルを連続した位置に再配置。しかも、ディレクトリを最適な位置に並べ変えてシークタイムを短縮。ディスクの高性能化とともに大幅なスピードアップを実現。まさに決定版のディスク・オプティマイザー。

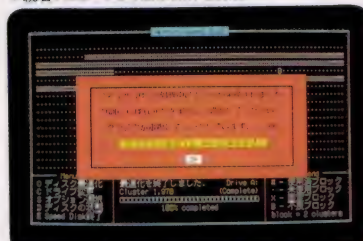
ノートン・ディスク・ドクター

フロッピーやハードディスクのあらゆるエラーを自動的に診断し、修復する革新のユーティリティ。ブートレコード、FAT、ディレクトリ構造全体にわたって診断し、問題箇所を修復。レポート作成や診断結果の表示も行なえるなど他と一線を画す高度なユーティリティ。

ノートン・ユーティリティ

多彩な先進機能を備えたメインプログラム。例えば、消去したデータの復旧機能。単独でリカバー能力を持ち、ファイル名やディレクトリ構造を変えない、他に類を見ない一級の機能。しかも、データ・フォーマットを問わずにディスク上のすべてのエリアを探索、編集できます。

●統合メニューにより簡単操作を実現

●強力な最適化機能により、ディスクアクセスを高速化

●強力なエラーチェック機能によりデータ破壊を防止

●初心者にも使いやすく設計されたオンラインヘルプ機能

ユーティリティの世界を革新した男、ピーターノートン。伝説のファイルリカバリツールUnEraseとともに彗星のように登場。以来わずか数年で米国PCソフト市場に不動の地位を確立。業界に大きな発言力を持つ権威者です。ノートン・ユーティリティーズ・アドバンスト・エディションは、最新の製品。全米トップテンの常連であり、市場全体を大きくリードしています。

"Peter Norton"

セーフ・フォーマット

MS-DOSのフォーマットコマンドを強化・拡張するユーティリティ。より安全、スピーディに、しかも簡単にディスクをフォーマット。誤ってフォーマットした場合でも以前のデータをフォーマットリカバーを使って簡単に復旧できます。

パッケージにはディスクのABCを解り易く解説したノートンディスクコンパニオンが付属。また、より高度なディスク保守を可能にする手引書ノートントラブルシューター。トラブルシューターはユーザー登録された皆様にもれなく贈呈。

ワイプ・ディスク／ワイプ・ファイル

米国国防総省(ペンタゴン)の最新データセキュリティ基準「DOD 5220.22M」をサポート。データの機密保全を行うまさに先進のユーティリティ。

さらに、珠玉の21機能

ノートン・ユーティリティーズ・アドバンスト・エディションには、このほかにも多彩な21機能を装備。標準MS-DOSの各機能を大幅に拡張・強化するユーティリティをはじめ、ディスクやシステム状況の詳細なインフォメーションを表示する機能、チェック機能も万全です。多機能、多彩な全27ユーティリティがMS-DOSとハードウェアを極限まで活用します。

THE NORTON UTILITIES ADVANCED EDITION

イントロダクションセール実施中

新発売を記念して、8月1日から10月17日までにご注文いただいた方には、定価38,000円を特別価格20,000円にてご提供。(消費税別)

LET'S JOIN! ピーターノートン・クラブ

ぜひお薦めする、ピーターノートン・クラブへのご入会。最新製品情報のご提供やイベント開催へのご招待をはじめ、さまざまな会員特典をご用意。詳細は、事務局まで。

03-479-7151 PNクラブ係

対応機種―
PC9801 FM以降
(ただしLT・ハイレゾは除く)
〔本体メモリ640KB MS-DOS Ver 2.11以上〕

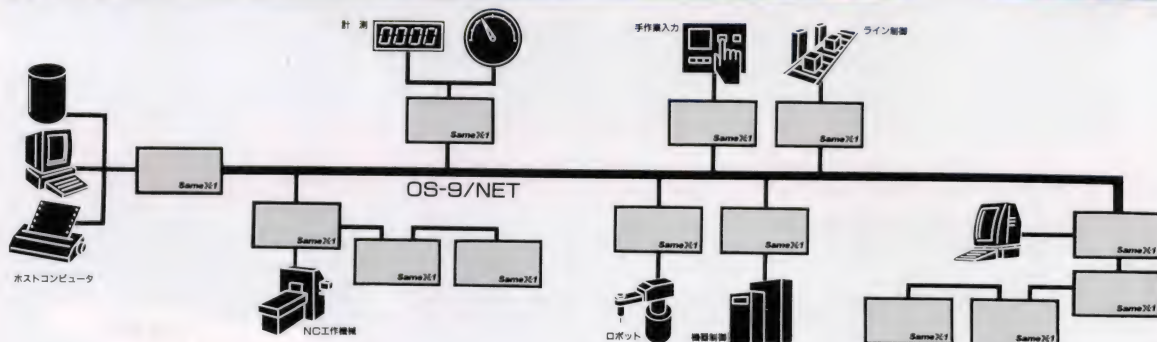
●お求めの際は、全国有名マイコンショップ、または直接弊社まで。

Peter Norton COMPUTING

ソフトウェア・インターナショナル株式会社
〒107 東京都港区南青山2-9-28S.I.ビル TEL.(03)479-7151代 FAX.(03)479-7158

資料請求No.66

HUMO

OS-9/LAN FAネットワークシステム ユニット

フレキシビルネットワークを徹底追求した ファクトリーLAN。

HUMO Same X-1は、OS-9/LANを使用したFA用ネットワークシステムユニットです。OS-9/LANの特徴を生かしてＦＡ用のラインシステムとしてインテリジェントされ、メインフレームに必要なI/Oを組み込む事により、容易にネットワーク制御機が構築出来ます。

〈特長〉

- 小規模なシステムから大規模なシステムまで柔軟に対応できます。
- OS-9/NETは、オンライン制御系のプログラムが非常にシンプルになり、強度なモジュール構造ができます。
- 各ノード間でのプログラム転送ができます。
- マルチウインドウ内での各ノードのモニターができます(shellの実行)。
- 各ノードのアプリケーションプログラムをホストコンピュータより変更ができます。
- レスポンスの良いLANの構築ができます。
- 30数種類の豊富なI/Oモジュールが準備されており、制御の対応が柔軟にでき、又ハンドラパッケージも準備されております。
- 低価格、高性能のユニット化で、コストパフォーマンスに優れています。

HUMO HUMO Laboratory, Ltd.
株式会社ヒューモラボラトリー
本社 〒167 東京都杉並区西荻北5-19-11 TEL.03-395-5311(代) FAX.03-395-5329

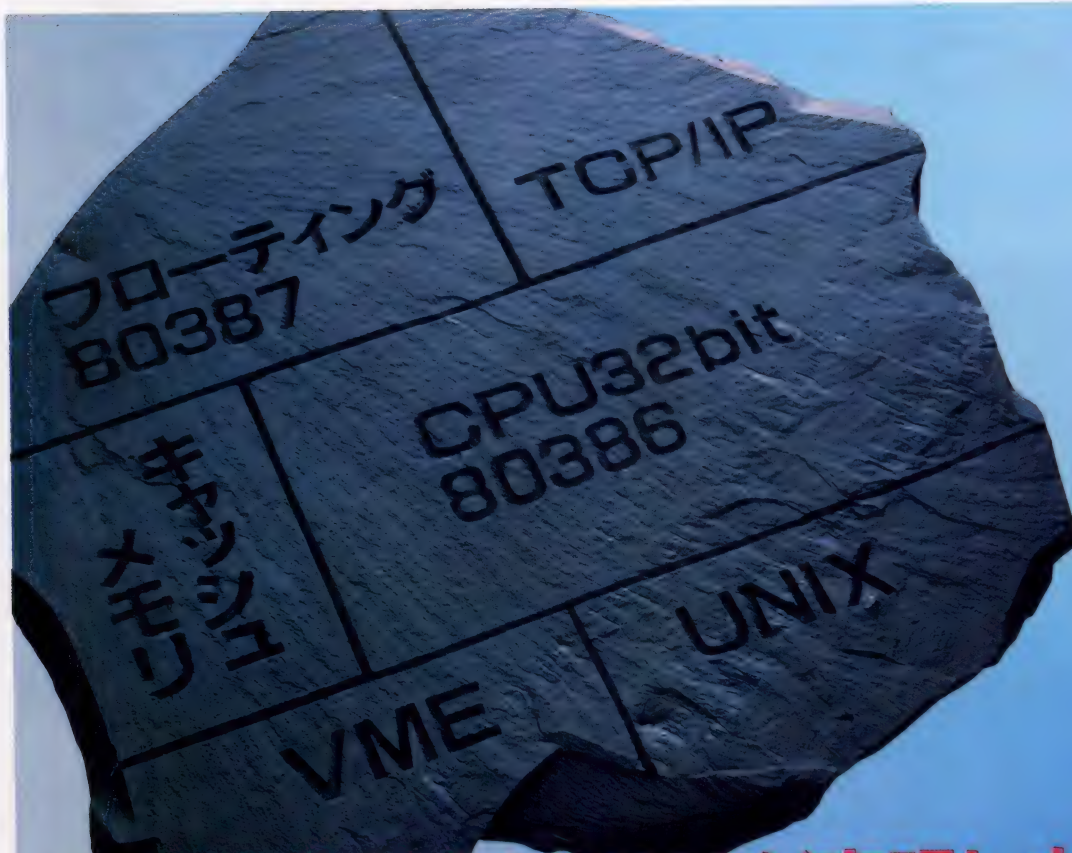
フローティング
80387

TCP/IP

キャッシュメモリ

CPU32bit
80386

UNIX

VME

1台からOEM。KX386が実現します。

さらに多彩に、さらにインテリジェントに ― 自在に組めるフレキシビリティーを備えた高機能ファクトリーコンピュータKX386。

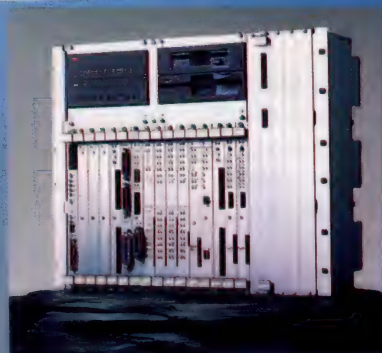
未来派エンジニア募集中

KX386は、CPUに80386、OSにUNIX、システムバスにVME busを採用したヘビーデューティー対応高機能ファクトリーコンピュータです。

生産現場の様々なニーズに応えるハード&ソフトの拡張性を備えています。

またVMEボードとしてはKINKEIが独自で開発したKVMEボードシリーズを提供。

多様化するシステムアプリケーションにフレキシブルに対応します。

■特長
- CPUに32bit80386を搭載、高い処理能力を誇ります。
- システムバスにVMEｂｕｓを採用、オープンアーキテクチュアを採り入れています。
- マルチプロセッサ方式を採用、高度な数値演算やリアルタイム処理能力を発揮します。
- システムニーズにフィットした補助記憶装置の選択が可能です。
- システムや使用環境に合ったケースの選択が可能です。
- OSにAT&T UNIX SystemⅤ系を標準装備、高信頼性を実現しています。
- X.25やイーサネットをベースにしたTCP/IPネットワークが構築できます。
- SCO Xenix SystemⅤ386により、豊富なプログラム開発ツールの提供が可能となります。

高機能工業用コンピュータ
KX386

アナログとデジタルを結ぶシステム計測
株式会社 近計システム
情報機器事業部

本社・営業本部：〒547 大阪市平野区加美北1-22-17 TEL(06)757-1321 FAX(06)758-4485
東京営業所：〒116 東京都荒川区東日暮里5-51-11(静屋ビル) TEL(03)803-4173 FAX(03)803-4168
福岡営業所：仙台営業所・本社工場・南港工場
創業・昭和8年8月/設立・昭和37年12月・資本金・8,437.5万円
営業項目・1)デジタル機器・コンピュータ周辺機器・その他電子機器の製造・2)電力線事故監視記録関係機器の製造・3)各種試験装置の製造

資料請求No.68

[Xのサーバ機能をパソコン上で実現。パソコンがUNIXのグラフィック・ターミナルに!!]

UNIXのグラフィック・インターフェースとしてすでに業界標準となっているX Window。アスキーは、MS-DOS、UNIX両面での技術の蓄積をフルに活用し、このX Windowのサーバ機能をパソコン上で実現するX-server／DOSを開発。X-server／DOSを使用することによって、従来、文字端末としてしか使用できなかったパソコンをUNIXマシンに対するマルチウィンドウのグラフィック・ターミナルとして使用することができます。

▼

X-server／DOSは、X Window Systemの最新バージョン(Ver.11 Release 2、3)のサーバ機能をパソコン上で実現した画期的な製品で、以下の特徴を持っています。

■カラー表示　RGBデータベースをサポート。ハードウェアの性能上可能な限りの色を同時に使用することができます(PC-9800シリーズの場合、4096色中16色の同時使用が可能)。

■日本語表示　ktermなどの日本語表示を行なうクライアントに対応。日本語表示用のフォントは、パソコン内部のROMを使用する省メモリ設計です。

■フォント・コンパイラ　フォント・コンパイラが提供されるため、*.snf形式で提供される標準フォント以外にも、BDF(Bitmap Distribution Format)形式で配布されるフォントを任意に追加することができます。

■LFD対応　Release 3で提案されたLFD(Logical Font Description)に対応。物理ファイル名に制限されることなく、フォントを物理名称で扱うことができます。また、フォント名の別名定義もサポートしています。

■X-server／DOS固有の機能　X-server／DOSでは、オリジナルのX11にはない、以下のサービスも提供されます。●MS-DOSコマンド・モードへの移行●rshサーバ機能のサポート●tftpサーバ機能のサポート●3ボタン・マウスのサポート●EMS(Ver3.2以上)に対応

パソコン対応高機能Xサーバ

X-server／DOS

*X-server／DOSは、86系のCPUとEthernetのネットワーク環境、MS-DOS(Ver.3.1以降)のシステム環境を前提としています。X-server／DOSは、OEM提供の商品です。現在エンド・ユーザの皆様には下記メーカーを通じてお届けしています。また、アスキーでは、OEM供給をお望みのメーカー各位からのお問い合せに積極的に対応しています。

日本電気株式会社　NECパソコンインフォメーションセンター　東京03(452)8000　大阪06(943)9800
ジャパンマクニクス株式会社　本社044(711)0330　大阪営業所06(300)0810

〒107-24東京都港区南青山5-11-5住友南青山ビル　㈱アスキー　システムソフトウェア事業部　営業部TEL.03(486)9075　**株式会社アスキー**

資料請求No.69

SOFT BANK

パソコンの基本、C言語はおもしろい。

Cプログラミング環境の新たなステージを切り開く。

C言語プログラマのための技術情報誌 月刊[Cマガジン]

C MAGAZINE 9月18日創刊

毎月18日発売・予価980円

『C MAGAZINE』は、"Turbo C" "Quick C" といった C コンパイラの登場によるプログラミングユーザーの拡大と、パソコンの機能向上によるハイレベルなプログラム環境の展開をとらえ、すべてのプログラミングユーザーが真に必要とする情報を提供していきます。

■創刊特別インタビュー①

C言語の立役者：Brian W. Kernighin

■第1特集

全コンパイラ完全フルテスト

■第2特集

C言語の光と影

——石田晴久・林晴比古が語るC言語の過去・現在・未来

■C言語入門講座

——はじめて学ぶCプログラミング

■実践的Cプログラミング入門

——開発者が語る実用プログラミングテクニック

■創刊特別企画

コンパイラの内部を徹底詳解

——文法解析・コード生成のしくみ

※内容は一部、変更になる可能性があります

［特別付録］
DISK版「C MAGAZINE」
掲載全ソースコード＋お楽しみプログラム

★売り切れる場合がありますのでお早めに書店でお買い求め下さい。

SOFT BANK 日本ソフトバンク出版事業部
〒102 千代田区九段南3-3-6
TEL：03-230-7670：営業局

信頼と実績で選ぶなら!

2500AD クロスツール

2500AD SOFTWARE INC.

Cコンパイラ

- 言語仕様K＆Rに準拠＋ROM化のための特別機能。
- ROM化の為に初期値付き変数の割付は、ランタイムアドレスを設定可能なインダイレクトリンク機能によりサポートしています。また、プログラムはコンパイラによって、PROGRAM、CONST、INT、UNINIT、の4種類に領域が自動的に分けられ、リンク時にユーザが割り付けることができます。
- 2500ADアセンブラにかかるアセンブラソースを出力するので、アセンブラソースレベルでのオプティマイズやチェックが可能です。
- Cのソースと対応したアセンブリソースの展開リストをとることが可能です。
- ランタイム用のスタートアップルーチン及び入出力ドライバモジュールはすべて、ソースコードで提供されますので各ターゲットへのインプリメントが容易です。
- インラインアセンブラ機能によりCソースプログラム内にダイレクトにアセンブラ記述が可能なのでタイミング制御等の微細な処理を可能にします。
- 強力なオプティマイザにより実行効率の高くコンパクトなコードが生成されます。
- C言語レベルでの割込み処理記述が可能です。
- ライブラリは完全なリエントラントであるため、リアルタイムシステムに使用可能。

アセンブラ

- 高速性を第一の主眼に設計されているため、アセンブルはすべてオンメモリで行なわれ、非常に高速なアセンブルが実現されています。
- アセンブルできるソースプログラムのサイズに制限はありません。
- アセンブル／リンク時のシンボル数、ファイル数に制限がありません。
- リストコントロール命令にはプログラムの全部、一部分またはアセンブリエラーのみをリスト出力させる等の重要な機能があります。
- 27種類の演算子を持ち、80ビット高速演算子をサポート、最大16個のオペランドを記述可能です。
- リテラル、ストリング、コメントに日本語(第1、第2水準漢字含む)が使用できます。
- 豊富な条件命令により構造化プログラミングが可能です。
- 強力なマクロ機能は、リカーシブ／ネストが可能で、C言語の構造体定義と同様なマクロを作成することができます。
- クロスリファレンスリストを生成できますので、各種変数管理及びデバック時の効率化がはかれます。
- ローカルラベルをサポートしますのでモジュール化設計を実現できます。
- 強力なオーバーレイマネージャを搭載していますので、高効率を実現します。
- リンカはすべてRAM上で走りますので、超高速リンクが実現されています。
- リンク可能なファイルサイズには制限がありません。
- リンカが扱うグローバルシンボル、エクスターナルシンボルの数に制限はありません。
- 各オブジェクトファイルは最大256個のユーザーセクションを処理可能です。
- リンカは最大256個の入力ファイルと256個のセクション名を処理できます。
- リンカは以下のシンボルファイルのフォーマットを出力することが可能です。
  - —INTEL-HEX
  - —バイナリー形式
  - —TEKTRONIX-HEX
  - —拡張INTEL-HEX
  - —MOTOROLA (S19、S28、S37)

シミュレータデバッガ

- シミュレータデバッガは、ターゲットCPUを完全にシミュレートするシンボリックデバッガです。レジスタ表示、フラグ機能、割込みステータス、ダンプ、逆アセンブル、シングルステップ、ブレークポイント設定等数多くの機能を有しており、効率の高いデバッグが可能です。
- ROM/RAM領域を、任意に設定できます。また、ROMへの書き込み、メモリ外アクセス等の厳格なチェックを行ないます。
- ブレークポイントは16個の箇所に設定でき、パスカウンタも自由に設定できます。
- ポートを疑似的に、ファイルに割り当て、I/Oシミュレート。
- ハードウェアリセットによるコールドスタートは任意に設定可能で、リセットシミュレート機能が自動的に働きます。
- 割込みのシミュレーションは、ハードウェア割込み、ソフトウェア割込み両方をサポート。割込みのタイミング、処理プロセス等も任意に設定可能です。

■シミュレータデバッガの画面(64180)

```
2500 A.D. Software 64180 Simulator/Debugger (C) 1988 - Version 4.03a
A  : a0   F  : 00   IX      : 0000   PC : 00:000c  IN0   A,(34h)
B  : 00   C  : 00   IY      : 0000        00:000f  OR    02h
D  : 00   E  : 00   SP      : f04a        00:0011  OUT0  (34h),A
H  : 00   L  : 00   I       :   a0        00:0014  LD    A,00h
                    R       :   05        00:0016  LD    (f006h),A
A' : 00   F' : 00   CNTLA0  :   10        00:0019  LD    (f006h),A
B' : 00   C' : 00   CNTLA1  :   00        00:001c  EI
D' : 00   E' : 00   CNTLB0  :   07        00:001d  LD    A,(f005h)
H' : 00   L' : 00   CNTLB1  :   07        00:0020  SUB   00h
                    STAT0   :   02
S  Z  H  P/V  N  C  STAT1   :   02     MODE    IFF1    IFF2
0  0  0   0   0  0  TDR0    :   00        0       0       0
00:0000 : 31 da f0 3e  a0 ed 47 3e  c0 ed 39 33  ed 38 34 f6  1..>..G>..93.84.
00:0010 : 02 ed 39 34  3e 00 32 05  f0 32 06 f0  fb 3a 05 f0  ..94>.2  2  .:..
00:0020 : d6 00 28 f8  f3 ed 2b 00  00 18 45 21  00 f0 11 08  ..(...+....!....
SINGLE STEP          SKIP             MACHINE CYCLES                14
Enter Command :
```

■2500AD CROSS series価格表

ビット	ターゲット CPU	Cコンパイラ MS-DOS	アセンブラ MS-DOS	シュミレータデバッガ MS-DOS
32	68020	¥358,000	¥158,000	(¥158,000)
	32000	(¥358,000)	¥158,000	PLANNING
16	Z280	¥332,000	¥132,000	(¥132,000)
	8086/88	PLANNING	¥68,000	PLANNING
	80186/286	PLANNING	¥98,000	PLANNING
	V20/30	PLANNING	¥98,000	PLANNING
	68000/8/10	(¥332,000)	¥132,000	(¥132,000)
	H16	(¥332,000)	(¥132,000)	(¥132,000)
	Z8000	(¥332,000)	¥132,000	PLANNING
	65C816	(¥332,000)	¥132,000	PLANNING
	8096	(¥298,000)	¥98,000	(¥98,000)
8	Z80	¥298,000	¥98,000	¥98,000
	Z Super 8	(¥298,000)	¥98,000	¥98,000
	Z8	(¥298,000)	¥98,000	¥98,000
	64180	¥298,000	¥98,000	¥98,000
	8080	PLANNING	¥98,000	¥98,000
	8085	(¥298,000)	¥98,000	¥98,000
	8048	—	¥98,000	¥98,000
	8044/51/52	¥298,000	¥98,000	¥98,000
	80515	¥298,000	¥98,000	¥98,000
	8400/84C00	PLANNING	¥98,000	PLANNING
	83C351	¥298,000	¥98,000	¥98,000
	6800/2/8	(¥298,000)	¥98,000	¥98,000
	6801/3	¥298,000	¥98,000	¥98,000
	6804	(¥298,000)	¥98,000	PLANNING
	6805	(¥298,000)	¥98,000	(¥98,000)
	6809	¥298,000	¥98,000	¥98,000
	68HC11	¥298,000	¥98,000	(¥98,000)
	H8	(¥298,000)	(¥98,000)	(¥98,000)
	NSC800	¥298,000	¥98,000	¥98,000
	6301/03	¥298,000	¥98,000	¥98,000
	6501/11	(¥298,000)	¥98,000	PLANNING
	6502	(¥298,000)	¥98,000	¥98,000
	1802	—	¥98,000	—
	F8/3870	—	¥98,000	—
	7500	—	¥98,000	—

●Cコンパイラの価格はアセンブラ、リンカ、ライブラリアン、ライブラリを全て含んだセット価格です。●アセンブラはリンカ、ライブラリアンを含んだセット価格です。●( )は、開発中。

ICE対応コンバータ(無償)

- STCA アドテック製ICE
- STCS ソフィア製ICE
- STCU ユニダックス製ICE
- STCI 岩崎技研製ICE
- リンクオプション ザックス製ICE
- リンクオプション コアデジタル製ICE

完全日本語マニュアル

- アセンブラ 日本語マニュアル MX○○○○ ¥5,000
- Cコンパイラ 日本語マニュアル MC○○○○ ¥10,000
- シミュレータデバッガ MS ○○○○ ¥5,000

VAX用は、お問い合せ下さい。

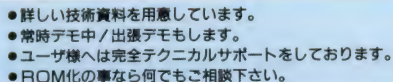
〔OS〕MS-DOS Ver. 2.x 3.x
〔必要メモリ〕512KB

※価格に消費税は、含まれておりません。

INDUSTRIAL SOFTWARE

- 詳しい技術資料を用意しています。
- 常時デモ中／出張デモもします。
- ユーザ様へは完全テクニカルサポートをしております。
- ROM化の事なら何でもご相談下さい。

株式会社エーアイコーポレーション
〒141 東京都品川区西五反田7-25-3 THビル TEL.(03)493-7981(代)

資料請求No.72

リンカで悩んでいませんか？

CrossCode C 68K

68000/10/20/30+68881/2
完全ANSI対応
クロスCコンパイラパッケージ

SOFTWARE DEVELOPMENT SYSTEMS INC.

リンク時に自由に領域を割り付けることができなくて、困っていることはありませんか？　ハードウェアを今更変えることもできないし．．．クロスCコンパイラパッケージの中で以外と大きな落し穴は“リンカ”です。コンパイラのコード生成効率や、デバッグ環鏡ばかりが、チェックポイントではありません。Cコンパイラやユーザが定義する領域を**自由自在に**、**制限なく**、任意のアドレスにリロケートできる、ということは当り前のことですが、これができない、あるいは制限が非常に多いコンパイラも存在しています。CrossCode C68Kのリンカは100%ユーザの意図するようなメモリマップでリロケーションを行うことができます。

■リンク後のマップ例

PARTITION	OVERLAY	REGION	BASE	SIZE
ROM	o1	reset	0H	8H
ROM	o1	vects	8H	0H
ROM	o1	code	2000H	100H
ROM	o1	const	2100H	0H
ROM	o1	string	2100H	0H
RAM	o2	data	C000H	64H
RAM	o2	ram	C064H	8H
RAM	o2	malloc	C06CH	0H
STK	03	stack	1FFFFH	0H

■リンクスペックファイル例

```
partition { overlay {
    region {} reset[addr=0];
    region {} vects[addr=8];
    region {} code[addr=0x2000];
    region {} const;
    region {} string;
    DATA = $;
} o1; } ROM;

partition { overlay {
    region {} data[roundsize=4];
    region {} ram[roundsize=4];
    region {} malloc[size=0x1000];
} o2; } RAM[addr=0xC000];

partition { overlay {
    region {} stack[size=0x1000];
    STKTOP = $;
} 03; } STK[addr=0x1ffff];
```

Cコンパイラ

- **完全ANSI規格**のシンタックスですから、DOS上のMicrosoft-CなどのCソースを有効利用することができます。
- **強力なオプティマイズ**により最小、最高スピードのオブジェクトが得られます。コンパイルスイッチにより、68000／20／30それぞれの拡張命令を利用した最適化を行います。レジスタ変数は16個まで使用可能です。
- 可変データタイプ
- lint**なみ**の強力なシンタックスチェック、タイプチェックをコンパイラオプションの指定で行うことができます。
- ライブラリは**全ソース**で供給。printf、scanfなどは低レベルルーチンをカスタマイズすることにより、そのまま使用可能です。
- 直接ハードウェアにアクセスする部分など**インラインアセンブラ**で記述することができます。
- **Cソースと対応するアセンブラ展開のリスト**を生成可能。
- **スタートアップルーチン**―初期値なし変数の初期化、初期値付き変数のRAM転送など、必要な初期設定をすべて行うスタートアップルーチンが用意されているので、ユーザが書く必要はありません。

アセンブラ

- **Motorolaアセンブラとの完全互換性**があるので、既存のアセンブラソースをそのままアセンブルすることができます。
- **強力なマクロ**はマクロ内条件分岐命令、パラメータのマクロ化などもサポートしています。
- **制限なし**＝シンボル数、ファイル長、ストリング長、includeのネストの深さ、などに制限はありません。どんなに大きなプログラムでもアセンブル可能です。

リンカ ＆ ダウンローダ

- 領域の割り付けは**100%ユーザの自由**になります。
- **Cのタイプチェック**をリンク時にも行えます。
- **マルチプルオーバレイ**をサポート。複雑なオーバレイにも対応。
- **インクリメンタルリンク**により、リンク後のファイルの再リンクが可能。
- **制限なし**＝オブジェクトサイズ、オブジェクトファイル数、シンボル数、などに制限がありません。
- 出力フォーマットは、Motorola-S、Intel-Hex、Tek-Hex、拡張Tek-Hex、等主要なフォーマットすべてを生成可能で、またユーザが定義する独自フォーマットで出力することもできます。
- リンク時に**新たなシンボルを定義**することが可能です。
- ROMへの**バイトスプリット**を8、16、32ビットで行うことが可能。
- 出力ファイルには**パーティション**を指定できるため、メモリセクションの任意の名前で必要部分だけ出力することができます。

ＩCEへの対応

- ソフィア、岩崎技研、ZAX、CORE、R＆D、HP、アプライドマイクロ等のICEに直接読み込めるシンボルファイルを作ることができるので、シンボリックデバッグが可能です。

C行番号のシンボル化

- Cの行番号をシンボルファイルに出力させることができます。行番号は“L＿123”のようなローカルシンボルとして生成されます。行番号によるブレーク機能があるICEにロードすることによりラインブレークをすることが可能。

充実したユーティリティ群

- **ライブラリアン**…ライブラリの製作、保守、管理を行います。
- ABS…アブソリュートリストを作ります。デバッグに大変便利です。
- SYM…さまざまなスタイルでのシンボルレポートを行います。
- GXREF…グローバルクロスリファレンサ。
- LINKPOST…ファイルシュリンカー、ファイルサイズの縮小ユーティリティ。
- SORTER…ソートユーティリティ。

〔OS〕MS-DOS　Ver. 2.x　3.x
〔必要メモリ〕512KB　〔価格〕¥398,000

※価格に消費税は、含まれておりません。

- 詳しい技術資料を用意しています。
- 常時デモ中／出張デモもします。
- ユーザ様へは完全テクニカルサポートをしております。
- ROM化の事なら何でもご相談下さい。

株式会社エー・アイ・コーポレーション
〒141 東京都品川区西五反田7-25-3 THビル TEL(03)493-7981(代)

資料請求No.73

新★登★場

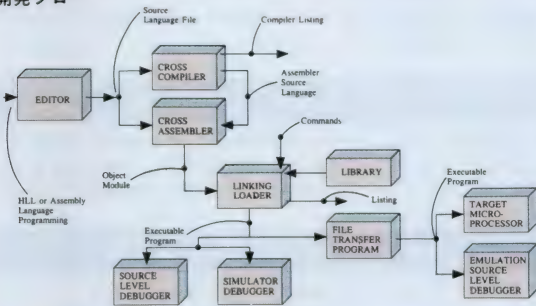
SEL H8/H16 Cコンパイラ

SOFTWARE ENVIRONMENT LTD.

AICはROM化ツールスペシャリストとして、ROM化開発をされている技術者の方のさまざまな要求に合うツールを提供するために、海外よりいろいろな製品をラインアップして開発者の皆様の選択の幅を広げてゆくよう努力しております。今回、アメリカのSEL社の高性能ROM化用クロスCコンパイラの発売を開始いたしました。

Cコンパイラ

- 1パスの高速クロスCコンパイラでアセンブリソースを出力します。
- H8/H18がターゲットとするリアルタイム制御用に耐えうる小さく、速いオブジェクトコードを生成するために強力なオプティマイズを行います。
  - ― 共通サブイクスプレッションの除去　　― レジスタトラッキング
  - ― ロジカルショートサーキット　　― レジスタカラーリング
  - ― 定数のフォルド　　― ループインバリアンス
- ANSI最新規格、"K＆R第2版"に完全対応のシンタックスで、MS-DOS上のMicrosoft-Cなどのソースプログラムをほとんどそのままコンパイルできる移植性に優れたシンタックスになっています。
- コンパイル時のリスティングはCソース行と対応するアセンブリソースを出力するリストを生成可能です。また、リンク後には絶対アドレスの付いた、アブソリュートリスティングも生成可能で、これは有効なデバッグ補助手段になります。
- "Signed" "Unsigned"インテジャータイプ(8、16、32ビット)をサポートしています。
- Cコンパイラは制御用の割り込みシステムやリアルタイムマルチタスクシステムに対応できるように完全にリエントラントなコードを生成します。
- 制御用に不可欠のインラインアセンブラ記述を行うことができます。

開発フロー

アセンブラ

- マクロ機能をもった高速クロスアセンブラで、H8／H16のニーモニックとアドレッシングモードを完全にサポートしています。
- IFEQ、IFNE、IFGT、IFGE、IFLT、IFLE、AELSE、ENDCなどの条件アセンブリ疑似命令が豊富に用意されています。
- 次のようなストラクチャードアセンブリ疑似命令により、高級言語なみの構造化プログラミングを行うことができます。ネストは16レベルまで可能です。

```
FOR     ―  ENDF   ―  ENDI
IF      ―  ELSE
REPERT  ―  UNTIL       WHILE    ―  ENDW
SWITCH  ―  CASE    ―   DEFAULT  ―  ENDS
BREAK      CONTINUE
```

オブジェクトフォーマットユーティリティ(OFU)

- オブジェクトファーマットユーティリティはリンカ、ライブラリマネージャ、その他ROM化に必要なユーティリティを完全にそろえたパッケージです。
- リンカはコンパイラが出力したコード、データ、ストリング領域やユーザが定義した領域をCPUのメモリ空間に自由に割り付けることが可能です。
- ライブラリアンは、ライブラリの作成、管理などのマネージングを行います。
- リンク後にはICEやROMライターにダウンロード可能な以下のようなフォーマットのファイルを生成することができます。

Motorola-S,　TEK-HEX,　日立SYSROF,
Intel-Hex,　拡張TEK-HEX,　HP64000

- OFUは、外部シンボルリスト、セクション情報リスト、メモリマップ情報、アブソリュートリスト、シンボルクロスリファレンス、ファイル使用情報、エラーサマリ、ユーザ定義メッセージなどの様々なリストやシンボルファイルを生成することができます。

ライブラリ

- IEEE規格の浮動小数点演算、超越関数、算術関数を完全にサポート。
- インテジャー、フローティング、各種データタイプのフォーマット化入／出力関数(printf, scanfのような関数)も用意しており、ユーザは低レベルの入出力部のルーチンを書くことによりカスタマイズすることができます。
- 別売でライブラリソースも用意しています。

ROM化のための特別仕様

SEL-CコンパイラはROM化のためにいろいろな工夫をしています。

■スペシャルメモリキーワード

- _basepage…宣言したデータを256以下のアドレスに配置する
- _wordpage…宣言したデータを0－65535の間のアドレスに配置する
- _alternate…CPUのアドレス範囲ならどこでも宣言したデータを配置する
- _at…指定したアドレスに宣言したデータを配置する

〔例〕 _basepage("video_ram") Char screen[200]
配列screenを領域"video_ram"に格納
_at (0x100)　char　INPORT;
_at (0x101)　char　OUTPORT;
変数"INPORT"をアドレス0x100に、変数"OUTPORT"をアドレス 0x101に格納する

■スペシャルモディファイヤ

- _interrupt…割り込みサブルーチンを関数として記述する

〔例〕
```
extern   zap();
_at(0x100) char port; /* I/Oポートの定義 */
main()
   {
   signal(zap.SIG_IRQ); /* 割り込みベクタの設定*/
   while(1);            /* 無限ループ */
   }
_interrupt zap()
   {
   port=0;              /* 割り込みをクリア */
   }
```

- _protect…割り込み禁止関数

(OS) MS-DOS (必要メモリ) 512K

(価格) Cコンパイラの価格はアセンブラ、OFUを含んだ完全なセット価格です。アセンブラの価格はOFUを含んだセット価格です。

※価格に消費税は、含まれておりません。

価格	Cコンパイラ	アセンブラ
H8	￥420,000	￥175,000
H16	￥480,000	￥250,000

★モニター募集中(モニター特別価格があります)

- 詳しい技術資料を用意しています。
- 常時デモ中／出張デモもします。
- ユーザ様へは完全テクニカルサポートをしております。
- ROM化の事なら何でもご相談下さい。

株式会社エーアイコーポレーション
〒141 東京都品川区西五反田7-25-3 THビル TEL(03)493-7981(代)

Microsoft-CのROM化2手法！

C_LOC & ROM-DOS

DataLight Inc.

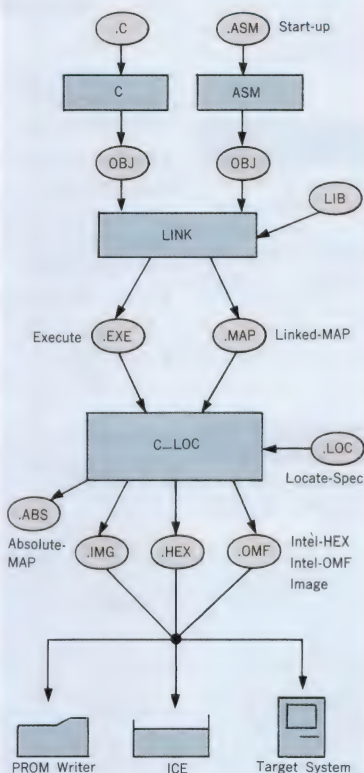
C_LOC
Microsoft-C用ROM化開発パッケージ

C_LOCはMicrosoft-Cを使ってROM化プログラムを開発するためのパッケージで、ROM／RAM領域への割り付けを行い各種ロードモジュールフォーマットを出力するロケータと、スタートアップルーチン、ROM化のためのノーハウなどが記述されたマニュアルから構成されます。

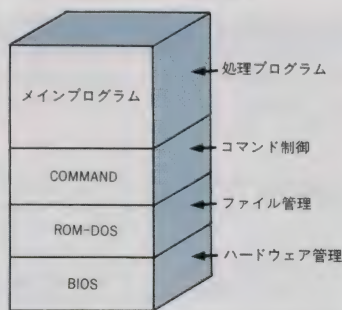
- ロケータはMS-DOSの実行ファイルを開発ターゲットハードウェアのメモリに再ロケートし、完全なROM／RAM領域別のマッピングを可能にします。
- ロケータが生成するロードモジュールは拡張Intel-HEX、Intel-OMF、バイナリイメージの3種類を選択できます。Intel-OMFでは出力モジュール中にデバッグ情報が出力されるので、ICEやシュミレータに読み込みシンボリックデバッグが行えます。
- Microsoft-Cのスタートアップルーチンは MS-DOS 環境を前提としているためにまったくROM化には適していません。C_LOCにはROM化用に特別に作られたスタートアップルーチンが用意されています。セグメントの定義／ROMからRAMへの初期値転送／初期値なし変数領域のゼロクリア／リセットベクタの設定、メインの呼び出し、リターン処理など組み込みに必要不可欠なソースコードが詳細に記述されています。
- マニュアルは完全日本語化しており、AICでさらにROM化ノーハウ情報を付加していますので、ROM化が初めての人にも簡単に開発を行うことができます。

開発プロセスフロー

.C → C → OBJ　.ASM (Start-up) → ASM → OBJ　→ LINK ← LIB
LINK → .EXE (Execute)、.MAP (Linked-MAP) → C_LOC ← .LOC (Locate-Spec)
C_LOC → .ABS (Absolute-MAP)、.IMG、.HEX、.OMF (Intel-HEX、Intel-OMF、Image)
→ PROM Writer、ICE、Target System

〔OS〕MS-DOS Ver. 2.x 3.x
〔価格〕¥98,000
※価格に消費税は、含まれておりません。

ROM-DOS
MS-DOS互換ROM化専用DOS

ROM化システムのプログラムは通常、DOSやI／Oシステムを独自に作成し、それをメインプログラムと共にターゲット上に組み込み動作させていました。しかしMS-DOSを、そのままターゲット上に置きMS-DOSのDOSコールやI／Oシステムのサービスを使えたらどれほどの開発コストと時間の節約になるでしょう。

- **米DataLight社のROM-DOSは、MS-DOS互換のシステム環境をターゲットシステム上に構築しユーザーアプリケーションを動作させます。**
- アプリケーションの組み込みは、MS-DOS標準のシステムコールが使用できます。
- BIOSは、容易にカスタマイズ可能。
- コマンドプロセッサの全ソースコードが付属。
- デバイスドライバの追加のみで、外部デバイスの追加／組み込みが可能です。

〔OS〕MS-DOS Ver. 2.x 3.x
〔価格〕¥100,000
※価格に消費税は、含まれておりません。

C_thru_ROM

C_thru_ROMはC_LOCにリモートソースレベルデバッガ(RDEB)を加えたMi crosoft-CのROM化開発デバッグ機能付きパッケージです。
C_thru_ROM を使うとつぎのような手順で開発を効率的に行うことができます。

1. MS-DOS上の実行プログラムを製作する。
2. コードビューでデバッグ
3. C_thru_ROMのスタートアップルーチンを使って再リンク
4. ロケータでROM/RAM領域別に再ロケート
5. 実機にデバッグ用カーネルと一緒にROM化
6. リモートソースレベルデバッガで実機上で最終デバッグ
7. すべてOKならばカーネルを除いて純粋にユーザプログラムだけをROMにプログラム

- リモートソースレベルデバッガは実機上でCレベルでのデバッグを行うことができます。このデバッガはMicrosoft-Cのコードビューと画面や機能をほとんど同じようにしているので、まるでMS-DOS上でのデバッグと同じ感覚で行うことができます。

リモートソースレベルデバッガはIBM-PC/AT、およびコンパチブル機、AXシリーズコンピュータ上で動作します。(PC9801シリーズ用開発中)

〔OS〕PC-DOS　〔価格〕¥220,000
※価格に消費税は、含まれておりません。

- 詳しい技術資料を用意しています。
- 常時デモ中／出張デモもします。
- ユーザ様へは完全テクニカルサポートをしております。
- ROM化の事なら何でもご相談下さい。

株式会社エーアイコーポレーション
〒141 東京都品川区西五反田7-25-3 THビル TEL(03)493-7981(代)

DOS Products

	Product	Price	Sys/Form/Req	Publisher
	AI EXPERT SYSTEM Dev't			
	Arity/SQL	¥58,500	MSD/5"	Arity Corp.
❖	CxPERT	¥115,000	MSD/5"	Software Plus
	Rule Master 3	¥150,000	PCD/3",5"	RADIAN Corp.
	ASSEMBLERS UTILITY			
	ASM FLOW	¥27,000	PCD/5"	Quantum Software
	Dis/Doc & Exe Unpacker	¥35,000	PCD/5"	R.J.Swantek
	Soft-X-plore	¥35,000	MSD/5"	R.J.Swantek
	Sourcer w/BIOS P.P.	¥35,000	MSD/5"	V Communications
	Unpacker	¥12,000	PCD/5"	V Communications
	Visible Computer 80286	¥23,000	MSD/5"	Software Master
	386:ASM/LINK	¥115,000	PCD/5"	Pharlap Software
	BASIC UTILITY			
	BASIC Development System	¥21,000	MSD/5"	Sterling Castle
	Graph Pak Professional	¥36,000	PCD/5"	Crescent Software
	Quick Menu	¥12,000	PCD/5"	Crescent Software
	Quick Pak Professional	¥36,000	PCD/5"	Crescent Software
	Laser Pak	¥16,000	PCD/5"	Crescent Software
	C UTILITY			
❖	Blackstar C Function Library	¥28,000	PCD/5"	Sterling Castle
❖	C Display Manager	¥36,000	PCD/5"	Sydetech
❖	C-Food	¥36,000 * W/S ¥72,000	MSD/5"	Lattice Inc.
	Cindex＋	¥78,000	MSD/5"	Trio Systems
	CLEAR + for C	¥37,000	PCD/5"	Clear Software
❖	C: LINES/C: TREE	¥18,000	PCD/5"	Soft Rex
New	C Programmer's Toolbox vol.1	¥23,000	PCD/5"	MMC Ad Systems
New	C Programmer's Toolbox vol.2	¥23,000	PCD/5"	MMC Ad Systems
New	C Programmer's Toolbox Combo	¥35,000	PCD/5"	MMC Ad Systems
❖	C TOOLSET	¥25,000	MSD/5"	Solution Systems
❖	Curses Screen Manager	¥36,000	PCD/5"	Lattice, Inc.
❖	C-Worthy	¥42,000	MSD/5"	Solution Systems
❖	Essential B-Tree	¥26,000 * W/S ¥50,000	PCD/5"	Essential Software
	Graphi C	¥98,000	PCD/5"	Scientific Endeavor
	Greenleaf Business Math Lib	¥87,000	PCD/5"	Greenleaf Software
❖	Greenleaf Data Windows	¥72,000	PCD/5"	Greenleaf Software
❖	Greenleaf Functions	¥53,000	PCD/5"	Greenleaf Software
	Greenleaf Make Form	¥29,000	PCD/5"	Greenleaf Software
	Greenleaf Super Functions	¥61,000	PCD/5"	Greenleaf Software
❖	PANEL Plus	¥115,000	PCD/5"	Roundhill Computer
	PC-lint	¥28,000	MSD/5"	Gimpel Software
❖	Screen Star	¥30,000	PCD/5"	Essential Software
❖	Sherlock	¥46,000	PCD/5"	Sherlock Software
	Stagehand	¥48,000	PCD/5"	Data Code
	Vermont Views	¥98,000 * W/S ¥170,000	PCD/5"	Vermont Creative
❖	Vitamin C with Source	¥50,000	PCD/5"	Creative Programming
	COMMUNICATION			
	ADComm	¥40,000	PCD/5"	Pinnacle Publishing
	ComPlus	¥15,000	PCD/5"	Pinnacle Publishing
❖	Greenleaf Comm Library	¥53,000	PCD/5"	Greenleaf Software
	Greenleaf View Comm.	¥120,000	PCD/5"	Greenleaf Software
	Side Talk	¥20,000	PCD/5"	Lattice, Inc.
	DBASE SUPPORT			
	CLEAR + for dBASE	¥37,000	MSD/5"	Clear Software
	Clipper	¥170,000	PCD/5"	Nantucket Corp.
❖	dBC III Plus	¥225,000 * W/S ¥450,000	PCD/5"	Lattice, Inc.
	dGE	¥44,000	PCD/3",5"	Pinnacle Publishing
	Front Runner	¥36,000	PCD/5"	Ashton-tate
	Genifer	¥85,000	MSD/5"	Byte Corp.
	GRW	¥50,000	PCD/5"	MIGHTYSOFT
	DEBUGGER			
	Breakout	¥35,000	PCD/5"	Essential Software
	Periscope I 512K	¥196,000	PCD/5"	Periscope Company

Look in OS/2 products

NeWS/2 OS/2用ネットワーク/拡張ウィンドウシステム

SUN-NeWSの強力なウィンドウ環境をそのままOS/2上で実現しOS/2プレゼンテーション・マネジャーに勝るプレゼン能力を持つ。SUN Microsystemsによって開発されたフレキシビリティ、移植性に富んだウィンドウシステム・プラットフォームNeWSのフルインプリメンテーション。PostScript言語を基調としており、PostScriptの洗練されたグラフィック機能を充分に堪能できる。
●サーバクライアント・モデルによる理想的なメモリ／ディスク資源運用●さらにはネットワークを介したマルチ／リモートクライアント運用が可能●ステンシル／ペイント・イメージング・モデリングにより、任意形状のウィンドウ生成や図形・文字の混在描画が可能●オペレーティングシステムやデバイスに依存しないアプリケーション記述が可能●ディスプレイ表示とページプリンタの出力には同一のNeWS環境により、SUNワークステーションでもPCでもアプリケーション開発が可能。UNIXからOS/2への単純な移植ではなく、更に以下の特徴がある●NeWS/2サーバは、LANマネジャー・ネットワーク上のどのマシンからでも「パイプ」を通じて自由にアクセス可能●NeWS/2セッションと、OS/2プレゼンテーション・マネジャーによる他のセッションは同時に実行可能●完全V10互換ウィンドウによりOS/2の通常のフルスクリーン・アプリケーションは、そのまま実行可能●NeWS/2のPostScriptシミュレーションにより、非PostScriptプリンタの使用も可能

OS2/3",5" ¥148,000 IMAGESOFT

Poly AWK パターンマッチング言語

Poly AWKはカーニハンらによって設計された"ツールボックス言語"。パターンマッチングを動作の基本としており、プログラマが殆ど定型的に要求されるデータ処理を簡単に記述でき、プロトタイピング・ツールとしても活用できる。オリジナルの言語仕様書である"The AWK Programming Language"が添付されている。

OS2/5" ¥50,000 Polytron

PVCS ヴァージョン・コントロール・システム

ソフトウェア開発が組織化・大規模化するにつれ、各工程で発生する様々なファイル管理の必要性とその複雑さは幾何級数的に増大する。その結果、プロジェクト・マネジャーやプログラマの貴重な時間がタイムスタンプの比較や変更箇所の追跡に費やされることになる。特に多人数でのプロジェクトや、LAN上での開発には効果的かつ統一された管理システムがいまや不可欠であり、突発的なファイル破壊に対する備えや、ファイル・アクセスに関するセキュリティ・チェックも必要となる。ソフトウェアの大生産国アメリカでは既にヴァージョン・コントロールの習慣が確立しており、その方法を統一するものとして数千にのぼる企業で採用されている標準システムが、PVCSである。●任意のヴァージョン、リビジョンの取り出し、格納：何世代か前に遡って任意のヴァージョンまたはリビジョンの取り出し、格納が可能●変更履歴のメンテナンス：PVCSのリビジョン管理が変更履歴により行われるため、世代の異なるヴァージョンまたはリビジョンの比較リストの出力、変更箇所の追跡が容易に行える●リビジョンツリー：特定のリビジョンから枝分かれさせ、複数のテキストを作成する場合の履歴管理を行うことができる●保護機能：プログラマ単位にパスワードを与えることで様々なレベルでのファイル・プロテクトを設定することができる●マージ機能：複数テキストから一つの新しいテキストをマージして作成する。Poly MAKE込。

OS2/5" ¥180,000 Polytron

Epsilon テキスト・エディタ

EpsilonはEELというCライクなプログラミング言語が組み込まれており、ユーザは好みの仕様のエディタを正確に作ることができる。また、並行処理の容易さが革新的な特徴であり、Epsilonの中からコンパイラ／リンカ／アセンブラや他のプログラムを走らせることができる。他のエディタと異り、並行してプログラムが走っている間、E-psilonによる編集作業を続けることができる。Epsilonは、大型機上のEMACSタイプのキー配列になっており、フル・マルチレベルundo/redoを供給する。IBM PC上で動作するエディタの中では最高速である。

OS2/3" ¥41,000 Lugaru

Vitamin C for OS/2 プロフェッショナル・ファンクション・ライブラリ

DOS版のVitamin C 3.0で使用可能な機能は全てOS/2版でも全て可能。Vitamin C for OS/2は、OS/2の公開された機能に依存しており、このことはユーザの開発するアプリケーションがOS/2プレゼンテーションマネジャー上でも動作することを意味する。主な特徴：多重オーバラッピング・ウィンドウズ、シングルフィールド、またはフルスクリーン・データエントリー・フォーム、無制限データ妥当検査、文脈依存ヘルプマネジャー、ロータス&Mac型メニュー、プログラム可能なキーボード処理、テキストエディタ・ルーチン、プリンタ出力ルーチン、ノーロイヤリティ、ライブラリソース無料。OS/2プロテクトモードと共に動作するが、通常のPC-DOS上でもコンパイル・リンクを行えるので、Vitamin C OS/2のみで両環境上でアプリケーションを動作させられる。MSC 5.1/Quick C用。

OS2/3" ¥70,000 Creative Programming

ADOS+ for OS/2 UNIXライク外部コマンド集

UNIXライクな外部コマンド集。フルスクリーンVi/EXエディタ、ファイルマネジメント・ユーティリティ、プリンタスプーリングなど28の外部コマンドを収容。

■収容コマンド

cal	ctags	kill	nice	rm	tee	vi
cat	diff	log	od	strings	touch	wc
cmp	grep	ls	pr	sum	utime	whereis
cp	head	mv	pwd	tail	version	which

OS2/5" ¥48,000 Max Ware

＊PCD=PC-DOS、MSD=MS-DOS、OS2=OS/2、W/S=with source.　＊PCD版は、IBM PC/AT、XT及び100％互換機でのみ動作します。
＊MSD版は、IBMをはじめとする一般MS-DOS上で動作可能ですが、日本語DOS上での動作保証は致しかねます。
＊予告なく価格が変更されることがあります。
＊当広告掲載製品のアフターサポートは弊社では致しかねますのでご了承下さい。

❖のマーク製品には( )内のLattice Cユーザ特価が適用されます。希望の方は注文書にLattice Cシリアル№をご記入下さい。

EDITOR			
BRIEF	¥42,000	PCD/5″	Solution Systems
EC-EDITOR	¥13,500	PCD/5″	C Soures, Inc.
ED Editing System Programmer's	¥86,000	MSD/3″,5″	American Cosmotron
Epsilon	¥41,000	MSD/3″,PCD/5″	Lugaru
KEDIT	¥39,000	PCD/5″	Mansfield Software
MKS VI	¥35,000	PCD/3″,5″	Mortice Kern Systems
MULTI-EDIT	¥28,000	MSD/5″	American Cybernetics
GRAPHICS			
dGE	¥44,000	PCD/3″,5″	Pinnacle Publishing
DESIGN CAD 3-D	¥80,000	PCD/5″	American small Business
EGA PAINT 2005	¥26,000	PCD/5″	Rix Softworks Inc.
Essential Graphics	¥65,000	MSD/5″	Essential Software
GFX Graphics	¥37,500	PCD/5″	C Sourse, Inc.
GFX Fonts & Menus	¥33,000	MSD/5″	C Sourse, Inc.
❖ Graphi C	¥98,000＊Personal use only	PCD/5″	Scientific Endeavors
Graphics and Animation Library	¥24,000	PCD/5″	Software Artistry
Graphmatic CGA & EGA	¥38,000	PCD/5″	Micro Compatible
Grasp	¥27,000	PCD/5″	Paul Mace Software
GSS Graphics Dev Toolkit	¥117,000	PCD/3″,5″	Graphic Software
❖ HALO '88	¥70,000	PCD/5″	Media Cybernetics
HALO Window Toolkit	¥CALL	PCD/5″	Media Cybernetics
PC Paintbrush Plus	¥24,000	PCD/5″	Z Soft
PC Paiutbrush Plus Window	¥25,000	PCD/3″,5″	Z Soft
GSS＊GKS for OS/2	¥120,000	PCD/5″	Graphic Software
Publisher's Paintbrush	¥48,000	MSD/3″,5″	Z Soft
STAT GRAPHICS	¥190,000	PCD/5″	STS C
LANGUAGE			
GAUSS	¥110,000	PCD/″5″	APTECH SYSTEMS
Go Script	¥55,000	MSD/3″,5″	Laser Go, Inc.
IntegrAda	¥130,000	MSD/5″	AETECH
Personal REXX	¥37,500	PCD/5″	Mansfield Software
Poly AWK	¥28,000	MSD/5″	Polytron
Topspeed Modula-2 & Techkit	¥45,000	PCD/5″	Jensen & Partners
Trilogy	¥33,600	MSD/5″	Complete Logic
Watcon C Optimizing Compiler	¥98,000	PCD/5″	Watcom Products
OS SUPPORT			
ADOS＋	¥27,000	PCD/5″	Max Ware
DESQview for PS/2	¥27,000	MSD/3″	Quarterdeck
DESQview API Toolkit	¥148,000	MSD/5″	Quarterdeck
Multi Boot	¥14,000	PCD/3″,5″	BOLT Systems
Peabody for MS-DOS	¥20,000	PCD,MSD/5″	Copia International
ZIP	¥220,000	PCD/5″	Binary Technics
TRANSLATOR			
BAS.C Commercial	¥85,000	MSD/5″	Gotoless Conversion
B Tran	¥120,000	PCD/5″	Software Translation
COMPILE 1-2-C	¥80,000	MSD/5″	Resource Analysis
New For_C w/s	¥300,000	PCD/5″	COBALT BLUE
PL/M to C	¥120,000	MSD/5″	Micro-Processor Services
SofTRAN	¥98,000	MSD/5″	Sulution Systems
TP2C	¥55,000	PCD/5″	BISS of Louisiana
80386			
DESQview API Toolkit	¥148,000	MSD/5″	Quarterdeck
FoxBASE＋Dev't Package	¥95,000	PCD/5″	Fox Software
SoftBytes 386max	¥25,000	MSD/5″	Microtek
VM/386	¥65,000	PCD/5″	IGC
386: ASM/LINK	¥115,000	PCD/5″	Pharlap Software
386: Debugger	¥50,000	PCD/5″	Pharlap Software
OTHER PRODUCTS			
AXOS	¥165,000	PCD/5″	Mc Soft Inc.
CLARION	¥198,000	PCD/5″	Clarion
COMPEDITOR II	¥70,000	PCD/5″	Ayeco
Dan Bricklin's Demo II	¥54,000	PCD/5″	Norton Computing
Data Shuttle	¥30,000	PCD/5″	Softway
Easy Flow	¥33,000	PCD/5″	Haven Tree Software
EVERTRAKER	¥70,000	MSD/3″,5″	Az-Tech Software
FASTBACK PLUS	¥30,000	PCD/5″	Fifth Generation
Flow Charting II＋	¥55,000	PCD/5″	Patton
Inside!	¥20,000	MSD/5″	Paradigm System
Instant Replay	¥37,000	PCD/5″	Nostradamus
Math Cad	¥78,000	MSD/5″	Math Soft
Mathematica 386	¥CALL	PCD/3″	Wolfram
MATRIX LAYOUT	¥35,000	PCD/5″	Matrix Software
MKS LEX & YACC	¥57,000	MSD/5″	Mortice Kern Systems
Norton Guides	¥30,000	PCD/5″	Norton Computing
Personal Measure	¥27,000	PCD/5″	Spirit of Performance
PolyDoc	¥58,000	PCD/5″	Polytron
PolyLibrarian II	¥45,000	MSD/5″	Polytron
PolySell	¥28,000	PCD/5″	Polytron
Sbrowse	¥130,000	MSD/5″	Paul Siegel Computer
Show Partner F/X	¥80,000	PCD/5″	Brightbill-Roberts
SPF/PC2.0	¥60,000	PCD/5″	Command Technology
STOP WATCH	¥15,000	PCD/5″	Custom Real-time Software
Virusafe	¥39,000	PCD/5″	COMETCO
Watch dog	¥85,000	MSD/5″	Fischer International
OS/2 Products			
ADOS＋	¥48,000	5″	Max Ware
C-Worthy	¥78,000	5″	Solution Systems
Command Line Editor	¥28,000	5″	Fountain Head
dBCIII Plus	¥225,000	5″	Lattice, Inc.
Epsilon	¥41,000	3″	Lugaru
Greenleaf Data Windows	¥98,000	3″	Greenleaf
Greenleaf Make Form	¥41,000	5″	Greenleaf
GSS＊GKS	¥120,000	3″	Graphic Software
GSS Graphics Dev. Toolkit	¥160,000	3″	Graphic Software
KEDIT	¥45,000	3″	Mansfield Software
Multi Boot	¥14,000	5″,3″	BOLT Systems
NeWS/2	¥148,000	5″,3″	IMAGESOFT
PC-lint	¥28,000	3″	Gimpel Software
PolyAWK	¥50,000	5″	Polytron
PVCS	¥180,000	5″	Polytron
Vitamin C with Source	¥70,000	3″	Creative Programming
Mac Products / UNIX Products			
CALL			

ここに掲載致しましたのはごく一部の製品です。毎月、最新情報を満載した総合カタログ『The Programmer's INDEX』をご送付致しますので、是非読者としてご登録下さい。

【登録方法】：官製葉書に ①ご氏名 ②ご送付先ご住所 ③お電話番号 ④年齢 ⑤特に関心がおありになる分野を明記して、下記までお送りください。

【ご注文方法】

ご注文書
- ご氏名
- ご住所（製品送付先）
- お電話番号
- 製品名
- DOS／ディスクサイズ
- 価 格
- ＋
- 消費税額（価格×0.03 1円未満切り捨て）
- ＝
- お支払金額

お支払いのご連絡
- ＋ 郵便振替の場合（東京6-352109）：入金済払込票のコピー
- ＋ 銀行振込の場合（第一勧業銀行、神田支店、普1297934）：払込金受取書のコピー
- ＋ クレジット・カードの場合（UC、VISA、JCB）：（申込書にカード№と有効期限を記入）

右記へ郵便またはFAXで

- ＋ 現金書留の場合：現 金

右記へ郵便で

＊当広告の表示価格は消費税を含んでおりません。お支払に際しては消費税額を含めてご送金下さい。

The Programmer's INDEX

a division of LIFEBOAT

TEL：03-293-3887
FAX：03-293-4710
■営業時間：月～金 9:00～17:30
■FAX受付：無休 24時間
〒101 東京都千代田区神田錦町3-6

C-Solution
キャンペーン

LIFEBOATはパワー

パソコン上でのアプリケーション開発状況は、現在、時代のニーズの多様化と共に大きく二極化されつつあります。ひとつの流れとして、ハードウェアの発展に呼応して、汎用的に拡大されたスペックを持つオペレーティング・システム上で、汎用アプリケーションを開発、または使用していく方向があります。それと同時に、特定機能のみ強化したアプリケーション開発ニーズの方向も、また、ひとつの流れとして、存在します。後者の流れの主流として現在、使い慣れたMS-DOSに、開発に必要な機能だけを拡張していくというパワフルなDOSの利用方法があります。LIFEBOATは、この様な現況をふまえて、特にC言語での開発場面に的を絞って、ユーザーのニーズに合った開発環境を、多岐にわたって提供致します。

MS-DOS機能を最大限に利用。効率よくアプリケーション開発が行えます。

MS-DOS専用Cコンパイラ

Lattice C/DOS Ver4.1

■レジスタ変数サポート
SI、DIレジスタをレジスタ変数に割当てます。従来のプログラムで、SI、DIレジスタを保存していないモジュールを利用するために、－giオプションで、サブルーチンコール前にSI、DIレジスタを保存するコードを出力できます。

■オーバーレイリンカ、ソースコード・デバッガ標準装備
Ver4.0以来のリンカ、デバッガをさらに強化。ソース・ウィンドウに、ソース・アセンブリ混合リスト、アセンブリリストも表示可能となりました。

■グローバル・オプティマイザ強化
不必要なループの最適化、未参照ローカル変数への代入の消去等、機能強化されました。

■最適化選択オプションの追加(速度vs大きさ)
デフォルトでは、速度を優先したコードを出力するようになっていますが、オプションにより、オブジェクトサイズを小さくするように最適化できます。

■キーワードの追加
●huge：32ビット・ポインタとして扱います。ポインタ計算時に正規化を行います。64KBより大きなオブジェクトを扱う場合に使われます。
●volatile：ほかのプロセス(ハードウェア割り込み)や、外部からの信号により変化する可能性のある変数を指定します。連続して参照する場合、レジスタから値をとらずに必ず、変数から値を獲得します。

■hugeモデルサポート
ヒープ領域以外に、64KBを越える静的データの定義が行えます。

■ビルトイン関数サポート
ビルトイン関数をサポートします。ビルトイン関数は、ライブラリからの関数コールではなく、その場でコード展開して実行されます。

ビルトイン関数一覧
abs　memcmp　memcpy　memset　strlen

これらの関数の使用頻度の高いプログラムはビルトイン関数を使用することにより処理速度が向上します。

■より実行速度の速いコード出力
ポインタ操作で、ロングポインタ(32ビットポインタ)の計算を行う場合、サブルーチン・コールによらず、32ビット演算コードを直接出力するようになったことにより、実行速度が向上しました。

Lattice C Ver J3.xのコード	Lattice C Ver J4.1のコード
XOR CX.CX MOV DX.0001 MOV AX.[BP+0C] MOV BX.[BP+0A] CALL (FAR) CXA38:アドレス計算 CALL (FAR) CXNM8:正規化	MOV AX.[BP+0C] MOV BX.[BP+0A] ADD BX.01 JNS LABEL AND BH.7F ADD AX.0800 LABEL:

●chip:constと同じように、定数を指定するキーワードです。constは、ANSI規格の定義に基づいているため、ROM化などの場合には機能的に不十分な場合がありますが、新キーワードchipによって、ROM化可能定数として定義できます。
●__interrupt:割り込み処理の関数宣言です。この宣言を持った関数は、自動的に全レジスタを退避、処理終了時に復帰します。これによって割り込みプログラムをすべてC言語で記述することもできます。

参考例
```
void    interrupt inter(es.ds.di.si.bp.sp.bx.dx.cx.
ax) short es.ds.di.si.bp.sp.bx.dx.cx.ax;
{    /*    割り込み処理    */
```

●align, noalign:オブジェクト別にword境界にそろえるかどうかを指定できます。

価格:¥98,000

対応機種
- ▶PC-9800シリーズ(LTではデバッガはラインモードのみサポート) ¥98,000
- ▶J-3100シリーズ(デバッガはラインモードのみサポート)……… ¥98,000
- ▶FM Rシリーズ　▶Panacomシリーズ……… ¥98,000
- ▶IBM PS/55……… ¥98,000
- ▶AX仕様機(デバッガはラインモードのみサポート)……… ¥98,000

Ver 4.0→4.1/DOS or /DUAL 無料バージョンアップ

OS/2もやっぱりLattice

MS-DOS/OS-2両用コンパイラ

Lattice C/DUAL Ver4.1

MS-DOS・OS/2用アプリケーションを、開発するためのコンパイラ。Lattice C Ver4.0のオーバーレイリンカが、DOSとOS/2の両用リンカに強化され、デバッガはCPR(CodePRobe)リアル・モード・デバッガと、CS(C-Sprite)プロテクト・モード・デバッガを標準備備しています。
pad, nopadキーワードによりプロテクト・モードでより効率的なデータ構造を、生成できるようになっており、グローバル・オプティマイザが生成コードを、更に最適化します。また、privateキーワードにより、Dynamic Link Libraryを生成する必要がある時、呼び出しルーチンのDSレジスタの値を保存して、内部のDGROUPセグメントに再設定するコードを生成できます。
Lattice C/DUALにより、MS-DOS上でもMS-OS/2上でも、OS/2アプリケーションを作成することができます。(本製品を動作させるには、MS-OS/2を必要とします。)

特　長

■レジスタ変数サポート
SI, DIレジスタをレジスタ変数に割当てます。従来のプログラムで、SI, DIレジスタを保存していないモジュールを利用するために、－giオプションで、サブルーチンコール前にSI, DIレジスタを保存するコードを出力できます。

■FAPIファンクションサポート
FAPI(ファミリ・アプリケーション・プログラム・インターフェース)ファンクション用のライブラリを標準装備していますので、FAPIとリンクされたアプリケーションは、MS-OS/2とMS-DOSの両方で実行することができます。

■グローバル・オプティマイザー標準装備
参照されていないローカル変数への代入などを消去します。

■5種類のメモリモデルをサポート
S(small), P(large program), D(large data), L(large), H(huge)の5種類のモデルをサポート。プログラムの大きさに合わせた最適なメモリモデルを選択することにより、効率のよいプログラムの作成ができます。

価格 ¥128,000

■対応機種
▶NEC　PC-9801シリーズ
リリース中　価格 ¥128,000
▶富士通 FM シリーズ
9月1日リリース 特価 ¥98,000
▶松　下 Panacom Mシリーズ
9月1日リリース 特価 ¥98,000

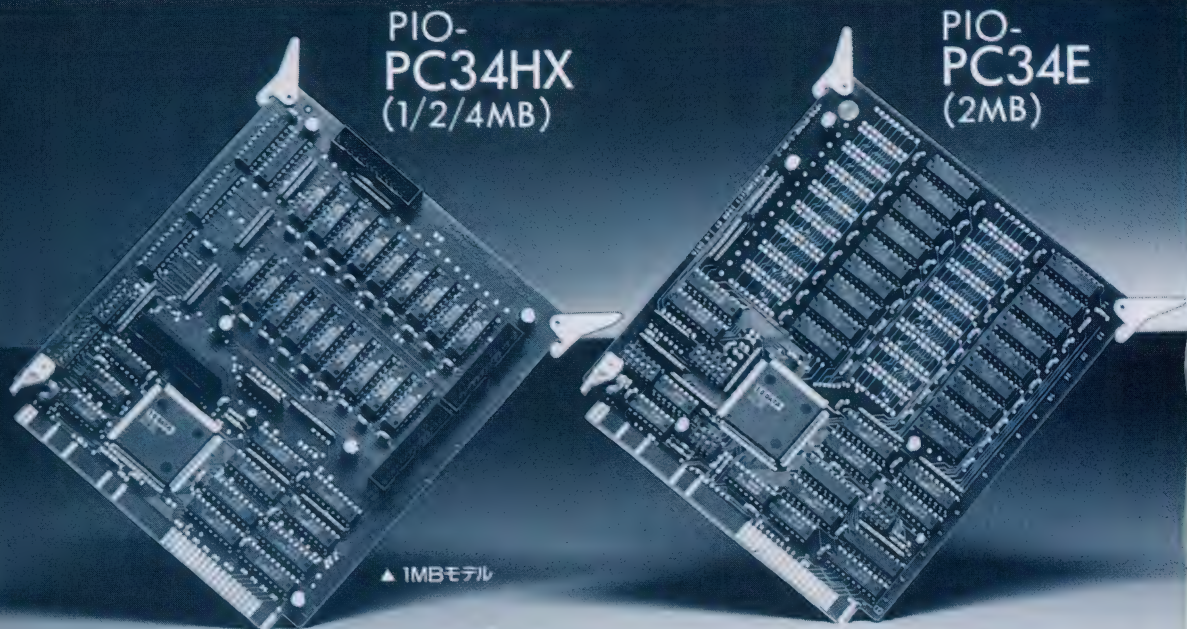
Host / Terget	Lattice C/DOS	Lattice C/DUAL
MS-DOS	●	●
MS-OS/2	×	●

Lattice C/DUALパッケージ内容
- ▶Lattice Cコンパイラ (DOS・OS/2両用)
- ▶Lattice LMBリンカ (DOS・OS/2両用)
- ▶Lattice CPRリアル・モード・デバッガ
- ▶Lattice CSプロテクト・モード・デバッガ
- ▶その他ユーティリティ

資料請求No.80

新世代“標準”

(6モデル)

EMSプラス増設RAMボードPC34シリーズ
(I・Oバンク/プロテクト)

90年代のパワーを秘めて——LIM 4.0 EMSを

PIO-PC34シリーズは、EMSモード、I・Oバンクモード、プロテクトモード、ハイレゾリューションの全モードに対応しています。さらに、EMS(EMSモード)とXMS(プロテクトモード)を同時に利用できるマルチモードを採用、MS-WINDOWS 2.1の機能をフルに活用できます。将来性を視野に入れながら、あらゆるユーザニーズに対応させ、信頼性、互換性、経済性を最優先させました。

■LIM EMS規格はすべて統一しています。
弊社LIM EMSメモリボードは全て統一された規格で設計されており、安心してご利用頂けます。もちろんEMSドライバソフトもIOS-10EMSで統一しています。

■高速ハードウェア処理のLIM 4.0 EMSモード対応です。各種EMS対応アプリケーションをフル活用できます
640KBの枠を超えるメモリ空間を利用するEMSモードでは、新設計ASICによる高速ハードウェア処理を実現しています。IOS-10EMS(別売)を利用して、各種EMS対応アプリケーションを効率的に利用でき、また、EMSモードでRAMディスクやディスクキャッシュ、プリンタスプーラを活用できます。
(EMS対応アプリケーション：EXCEL(マイクロソフト)、Lotus1-2-3(ロータス)、一太郎Ver4(ジャストシステム)他多数)

■実績のI・Oバンクモード対応です。BMS利用で各種BMS対応アプリケーションの機能をフルに発揮させます。
I・Oバンクモードに対応しているため、既存のI・Oバンクの資産が利用できます。さらに、IOS-10(別売)でBMS(I・Oバンクメモリ統一管理規格)も利用でき、各種BMS対応アプリケーションの高水準活用が可能になりました。
(BMS対応アプリケーション：八方美人2号、UP2シリーズ(ダイナウェア)、Recalc(アスキー)他多数)

■仕様

型式番号	PC34HX-1M	PC34HX-2M	PC34HX-4M	PC34H-4M	PC34H-6M	PC34H-8M
RAM容量	1024Kバイト	2048Kバイト	4096Kバイト	4096Kバイト	6144Kバイト	8192Kバイト
消費電流	460mA	470mA	480mA	480mA	485mA	490mA
供給電源	+5V±10%					
データチェック方式	パリティチェック					
対応モード	LIM4.0/3.2EMSモード/I・Oバンクモード/プロテクトモード/マルチモード/ハイレゾモード					
増設オプション	EX34-1M/2M	EX34-2M	EX34-4M	———	———	———

■対応機種

LIM4.0 EMSモード	PC-9800全シリーズ(PC-98XA、PC-98LT、PC-9801LV/LSを除く)(LIM4.0EMSではmax32Mバイトまで増設可能)
I・Oバンクモード	PC-9801/E/F/M/U/VF/VM/UV/VX/UX/CV/RA/RX/EX/ES、FC-9801/V/X/A ●リアルモード対応 ●メインメモリ容量512KBの時、増設可能 ●標準メモリ容量640KBのパソコン機種はメインメモリ容量を128KB切り離して増設(I・Oバンク方式増設RAMボードはmax32Mバイトまで増設可能)
プロテクトモード	PC-9801VX/UX/RA/RX/EX/ES、PC-98XA/XL/XL²/RL、FC-9801X/A ※パソコン機種により増設アドレスの上限は異なります

●PC34HX(1MB)…¥38,000/(2MB)…¥70,000/(4MB)…¥135,000 ●PC34H(4MB)…¥120,000/6MB…¥170,000/8MB…¥235,000
※増設オプション・EX34(1M)…¥35,000・(2M)…¥55,000・(4M)…¥110,000

I・O DATA パソコン増設RAMのトップブランド
株式会社アイ・オー・データ機器
本社/サポートセンター 〒920 金沢市駅西本町1-5-41 TEL.0762-21-4812 FAX.0762-24-9300
東京営業所 〒101 東京都千代田区神田富山町6 松崎ビル4F TEL.03-254-0301 FAX.03-254-9609

共通仕様(LIM EMS)で1MBから8MBまでフルラインアップ。

はじめとするメモリ拡張の全モードに対応。

■XMS*対応で、MS-WINDOWS2.1の高水準利用はもちろん、OS/2を使用する際のメモリ増設に利用できます。

1Mバイト超のアドレス空間の拡張メモリ仕様XMSで使用でき、MS-WINDOWS2.1はもちろん、MS-WINDOWS/386、OS/2を使用する際のメモリ増設に利用でき、将来的な活用にも対応できます。

＊XMSとはextended memory specificationの略です。

■高水準メモリ環境を実現するEMSモード＆BMSモード対応です。一枚のボードで両方のアプリケーションが併用可能です。

IOS-10EMSを利用すると、EMSモードでメモリウィンドウが16ページ同時に開けるため、一枚のボードでEMS(EMSドライバ)とBMS(BMSエミュレータ)が併用できます。バンクドライバ(FEPドライバ)など既存のBMS対応アプリケーションがそのまま利用でき、また、将来、同じOS上でEMSを使用するソフトとBMSを使用するソフトを動作させることも可能です。

■ここまで徹底、混在使用が可能。いっそう高度な活用を実現します。

既存のI·Oバンク、プロテクトRAMボードは、すでにPIO-PC34シリーズがEMSメモリとして増設してあるパソコンにそれぞれI·Oバンク、プロテクトメモリも混在使用可能です。さらに、I·Oバンク、プロテクトメモリをIOS-10、IOS-10PXでRAMディスク、ディスクキャッシュ、プリンタスプーラとして同時利用すればEMS対応アプリケーションのパフォーマンスをいっそう向上させることができます。

■うれしい倍速アクセス機構を搭載しています。特にEMSモードでは処理速度に大きな差がでます。

パソコン側の追加ウェイトサイクルをキャンセルすることによって、EMSメモリ、バンクメモリのアクセスタイムを短縮します。特に、EMSモードでは拡張メモリ上でプログラムが動作することが多いので処理速度に差がでます。

■必須、パリティチェック回路を搭載。EMSモード、プロテクトモードで、特に高信頼性を確保します。

EMSモード、プロテクトモードで必須のハードウェアによるパリティチェック回路を採用、高信頼性を約束します。拡張メモリ上でプログラムが動作するEMS、プロテクトモードでは、不可欠の機能といえます。

■ハイレゾリューション(NEC標準ウィンドウ方式)もサポートしています。

PIO-PC34シリーズは、PC-98XA/XL/XL²/RLのハイレゾリューションモードに対応しています。(PC34Hシリーズ)

■HXシリーズは拡張スロットを使用せずに容量を拡張できます。

HXシリーズは増設オプションのEXシリーズを装着することにより、拡張スロットを使わずメモリ容量を2倍に拡張できます。

サポートソフト

LIM 4.0 EMS仕様を完全サポート。PC34シリーズをマルチウィンドウワールドへ。

IOS-10EMS ■LIM EMSモード(LIM4.0/3.2 EMS) ■価格/¥5,000

BMSはプロテクトメモリもサポート。

IOS-10Ver.2.3 ■I·Oバンクモード/プロテクトモード(ハイレゾモードを除く) ■価格/¥5,000 バージョンアップサービス¥3,000

80386CPUのプロテクトメモリでEMS、BMSを実現。

IOS-10/386 仮想86モード用プログラム(近日発売)

IOS-10組込み相談室 ●東京営業所

好評発売中 I·O RAMボード

基本性能に徹した永遠のベストセラー	完全自動立上げを実現したスタティックの実力派	32ビット高速RAMの決定版
9234Gシリーズ (0.5MB〜16MB)	9834Lシリーズ (1MB/2MB/3MB/4MB)	RA34-3M (3MB)¥98,000

●32ビット98RL用高速RAM PIO-RL34-4M(4MB実装済) 近日発売
●FMファン待望FMR-50/60対応PIO-FM34発売中

※広告掲載の表示価格には消費税は含まれておりません。

マイコンシステムの教育ノウハウをここに結集

基礎講座の各コースで使用

ビデオ教材 マイクロコンピュータシステム開発基礎

特長

●基礎教育のためのハードウェア・ソフトウェア・学習手順が、これ一つですべてそろいます。●ビデオを見ながらテキストやトレーニングボードを使って体験学習ができます。●指導者用のインストラクターマニュアルが用意されています。

セット内容

●VTR(全4巻)●トレーニングボード(MTK8514、MTK8515)●テキスト、インストラクターマニュアル(各コース1冊)●トレーニングボード説明書(各1冊)●テスト●問題集●電源●テストROM●8085Aアセンブラ 他

セット価格 398,000円(税別)

メカトロニクス・エンジニアの育成に最適です

教育用ロボットLABOmarkII

特長

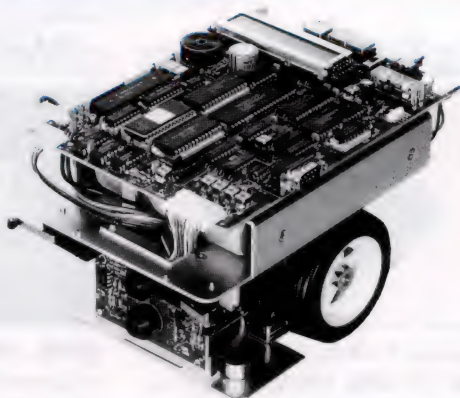
●パーソナルコンピュータからのリモートデバッグが可能です。●付属のソフトウェアによりユーザプログラムの開発をサポートします。●BIOSプログラムの内容をすべて公開しています。

ハードウェア構成

●CPU：M5M80C85AP(X'tal＝6.144MHz)●ROM：16KB●RAM：32KB(バッテリバックアップ)●拡張メモリ：8KB●周辺IC：M5M82C51AP、M5M82C54P、M5M82C55AP●RS-232Cインタフェース●ステッピングモータ×2●超音波センサ×1●赤外線センサ×4●LCD(20文字×1行)●スイッチ×2

セット内容

●本体●バッテリ●充電器●RS-232Cケーブル●プログラム開発用ソフトウェア

セット価格 195,000円(税別)

メカトロ&リアルタイム制御コースで使用

ワンチップマイコンの効率的な、『学習』『開発』環境を提供します

ワンチップマイコンMELPS740コースで使用

Designer's KIT 734

特長

●お買い上げいただいたその日からM50734の簡易型システム開発支援ツールが動きます。●ビジュアルなCRT画面表示とシンプルな操作性によって、コストパフォーマンスの高いデバッグ環境を提供します。●M50734を使用したシステムであれば、ハードウェアに依存しないモニタ。

セット内容

●テキスト●SRD734(リモートデバッガ)●RST734(ディスクベースのホストモニタ)●TRG734(ROMベースのターゲットモニタ)●ASM734(アブソリュート版アセンブラ)●MTK7408(CPUボード)●MTK7407(I/Oボード)●MTK7409(電源)●RS-232Cケーブル●40芯フラットケーブル●電源ケーブル

セット価格 48,000円(税別)

講習会・教材のお問合せはお電話で――

03-5261-0061 担当：松尾

Something New
いつも何かが新しい 三菱半導体

KMCが提案する最強の8ビット(Z80 64180 8085)環境!

クロスデバッガ
PARTNER-80

PC-98上でロジカルなデバッグをサポートするソースレベルデバッガ。

CP/M-80開発ツール
セルフコンパイラ(LSI-C HI-TECH C)

CP/M-80、MS-DOS統合環境
Turbo-V Turbo-180
MS-DOS上でCP/M-80 CP/M-86のプログラムを実行する為のエミュレータです。

セルフアセンブラ(PROASMII M-80, L80)

PC-9801
MS-DOS(Ver2、11、3.1、3.3)
メモリ 384K以上

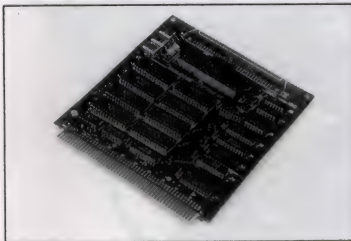
ターゲットデバッガ
PARTNER-T(80)

ターゲットシステムのROMソケットに接続して使うまったく新しいタイプの新世代ターゲットデバッガ。

MS-DOS開発ツール、ユーティリティ

クロスコンパイラ(LSI-C, HI-TECHC)
クロスアセンブラ(PROASM-II)
各種エディタ

近日発売

Turbo-V/180

Turboシリーズは、MS-DOS、CP/M86、CP/M80、CP/M68K(オプション)の4つのOSを融合した、すばらしいオペレーションおよび開発環境をユーザに提供するとともに、異種OSの障壁を克服して高速実行を可能とするためのOSアダプターです。

●MS-DOS用は、すばらしいMS-DOS環境を損なうことなく、MS-DOS, CP/M86, CP/M80, CP/M68Kのソフトウェアを自動判別して実行できます。もちろん、CP/M系のソフトウェアでもテンプレート,リダイレクト,パイプ,パス指定等を有効に活用できます。

●8ビットソフトは、V30CPU(8080エミュレーション)または、Z80上位コンパチブルHD64180 CPUによって超高速実行されます。

●CP/Mの各種のフォマットを読書きするためのメディア変換ユーティリティが付属しております。

価　格(消費税は含みません。)

	PC-9801/E/F/M	PC-9801VM以降
Turbo-V(CP/M86セット)	34,500	23,400
(MS-DOSセット)	34,500	23,400
(CP/M86+MS-DOSセット)	39,800	29,800
Turbo-180 6M	59,800	59,800
8M	69,800	69,800
超高速対応	79,800	79,800

PARTNER-T(80)

低価格、高機能、マルチCPU対応ソースレベルターゲットデバッカ。

●ROMソケットに接続して使うまったく新しいタイプの新世代ターゲットデバッガ

●対応するROMは2764、27128、27256、(27512)で2個、PARTNER-T2では4個

●サポートCPUは、Z80/64180/TMPZ84

●操作性に優れたウィンドウ指向デバッガ

●対応する処理系　LSI-C80, HI-TECH C80, PROASM-II, M-80など

●専用パラレルインターフェースによる高速で快適な動作環境

●ハードウェア ブレークとリアルタイム トレース機能(PARTNER-T2)

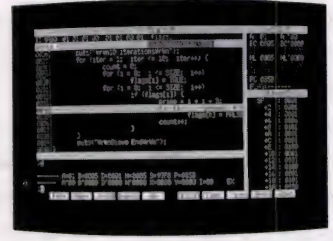
●強力なデバッグコマンドを装備

●デバッグを補助する1文字入出力システムコール

●ROMエミュレータとして使用するためのユーティリティソフト付属

●共通のハードウェアで86シリーズと80シリーズに対応(オプションコントロールソフトが必要。)

価格表(消費税は含みません)	
PARTNER-T1(86)、PARTNER-T1(80)(86又は80いずれかのコントロールソフトを含む)	各 198,000円
PARTNER-T2(86)、PARTNER-T2(80)(86又は80いずれかのコントロールソフトを含む)	各 298,000円
オプション コントロールソフト 86シリーズ用、80シリーズ用	各 98,000円

PARTNER-80

●MS-DOS上で8ビット用のソフトウェアのデバッグを行なうためのソースレベルクロスデバッガです。

●操作性に優れたウィンドウ指向デバッガ

●対応処理系　LSI-C80、HI-TECH C80、PROASM-IIなど

●180ボード及びV30の8080エミュレーション機能により高速実行

●Turbo-V/180と併用すればCP/M80用のプログラムデバッグも可能

PROASM-II

標準で7タイプのCPUに対応した、機能マクロ・クロス・アセンブラ!!

●マクロライブラリの作成により各種4ビット/8ビットCPUに柔軟に対応できます。

●標準仕様で、Z80・64180(647180)・8085・6809・8048・8051・6301(03)のアセンブラを装備。ニーモニックコードは、チップメーカー完全コンパチです。

●従来比3倍の高速アセンブル処理が可能。

●新発表のCPUにも、マクロライブラリの作成により、いち早く対応できます。

●各種OSに対応

〈価格〉 MS-DOS版　98,000円
(消費税は含みません。)

京都マイクロコンピュータ株式会社

〒617 京都府長岡京市長岡3丁目1-2
TEL(075)953-0963 FAX(075)953-0935

PARTNER-T(86)/(80)

高機能、マルチCPU対応ソースレベルターゲットデバッガ

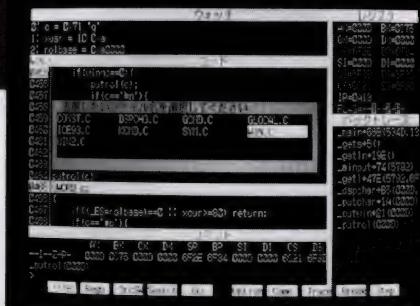
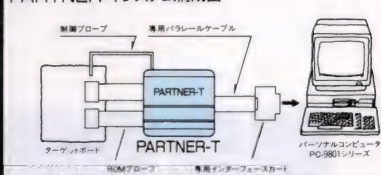
PARTNER-Tは独自の技術により従来のデバッグモニタとROMエミュレータを巧みに融合したハードウェアと、MS-DOS用デバッガ『PARTNER-S/H』で培ったソフトウェア技術をマッチングして誕生したまったく新しいタイプのターゲットデバッガです。

価　格　表(消費税は含みません)	
PARTNER-T1(86) PARTNER-T1(80) (86又は80いずれかのコントロールソフトを含む)	各198,000円
PARTNER-T2(86) PARTNER-T2(80) (86又は80いずれかのコントロールソフトを含む)	各298,000円
オプション コントロールソフト 86シリーズ用 80シリーズ用	各 98,000円

PARTNER-Tシステム構成図

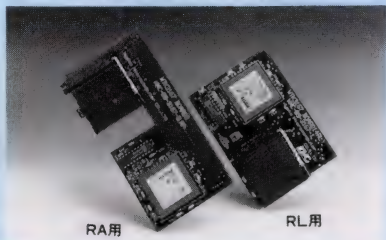
制御プローブ
専用パラレールケーブル
PARTNER-T
ターゲットボード
ROMプローブ
専用インターフェースカード
パーソナルコンピュータ
PC-9801シリーズ

- ●ROMソケットに接続して使うまったく新しいタイプの新世代ターゲットデバッガ
- ●対応するROMは2764、27128、27256、27512で2個、PARTNER-T2では4個
- ●サポートCPU
  8086 8088 80186 80188 80286(リアルモード) V20 V30 V40 V50 V25 V35(86シリーズ)
  Z80 64180 8085 TMPZ84(80シリーズ)
- ●操作性に優れたウィンドウ指向デバッガ
- ●対応する処理系
  MS-C, Turbo-C, Lattice-C, MASMなど(86シリーズ)
  LSI-C80, HI-TECH C80, PROASM-II, M-80など(80シリーズ)
- ●専用パラレルインタフェースによる高速で快適な動作環境
- ●ハードウェア ブレークとリアルタイム トレース機能(PARTNER-T2)
- ●強力なデバッグコマンドを装備
- ●デバッグを補助する1文字入出力システムコール
- ●デバッガ内で自由にMS-DOSコマンドを並行処理するマルチジョブ機能
- ●ROMエミュレータとして使用するためのユーティリティ ソフト付属
- ●共通のハードウェアで86シリーズと80シリーズに対応(オプションコントロールソフトが必要です。)

<動作環境>
- ●機種：PC-9801シリーズ（XA/XL/RLハイレゾ含む）
- ●メモリ：384Kバイト以上
- ●OS：MS-DOS Ver 2.11 Ver 3.1 Ver 3.3(86シリーズではMASMが必要です。)
- ●拡張スロット：1スロット使用

PARTNERシリーズマルチウィンドウ画面表示例

Turbo-3167RA/RL 新登場

超高速数値演算ボード

Turbo-3167は、PC-9801RA/PC-98RLでWEITEK社の高速数値演算プロセッサWTL3167を利用するための数値演算ボードです。

VAX8600並の高速処理

VAX8600並の高速処理および386プロテクトモードサポートの処理系により、4Gバイトダイレクトアクセス可能で非常に大きな配列の計算が実現できます。

●PC-9801＋Turbo3167を使用すればNDP FORTRAN386で記述された数値演算アプリケーションで2～3倍にパフォーマンスが向上します。

ベンチマークテスト

（機種PC-98RL 20MHz、コンパイラスイッチはフルオプティマイズ）

	PC-98RL＋387＋MS-FORTRAN	PC-98RL＋Turbo3167＋NDP FORTRAN	PC-98RL＋387＋MS-C	PC-98RL＋Turbo3167＋High C386
Whetstone (単精度)	936k Whetstones	2,650k Whetstones	883k Whetstones	2,564k Whetstones
500万回四則演算(単精度)	116.8秒	30.9秒	122.3秒	17.36秒
500万回四則演算(倍精度)	134.8秒	51.9秒	119.4秒	28.59秒

〈対応処理系〉
NDP C386 (Micro Way)
NDP FORTRAN386 (Micro Way)
High C386 (Meta Ware)
386ASM/LINK (Phar Lap Development Tools)

〈対応機種〉
PC-9801RA PC-98RL

価　格(消費税は含みません)	
Turbo-3167RA (PC-9801RA用)	338,000円
Turbo-3167RL (PC-98RL用)	338,000円

▲上記ソフトウェアは当社又はテックソフト & サービスにて取り扱っております。

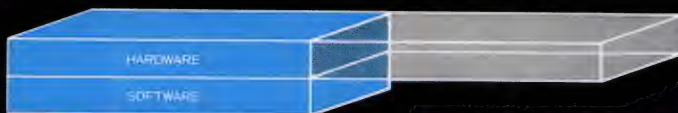
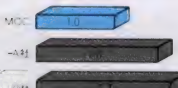
京都マイクロコンピュータ株式会社
〒617 京都府長岡京市長岡3丁目1-2
TEL(075)953-0963 FAX(075)953-0935

C言語シミュレートデバッガ

XRAY

VAX HP-9000
Sun apollo NEWS AS-3000 LUNA
J-3100 IBM-PC/AT PS/55 PC-9801 PC-286 AX

XRAY Z80・XRAY 86リリース開始!!

開発期間の大幅な短縮を実現。

お問い合わせは

〒604 京都市中京区蛸薬師通烏丸西入ル ヒライビル
TEL.075(211)1256 FAX.075(223)1095

多くのプラットホームで幅広いマイクロプロセッサをサポート

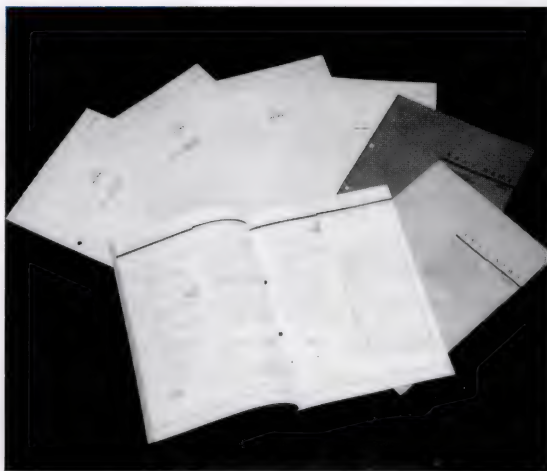
対象MPU	シミュレータバージョン	クロス・コンパイラ	クロス・アセンブラ
MC68000/10/20/30	XRAY68K	MCC68K	ASM68K
8086/88/186/188/286 V20/V30/V40/V50	XRAY86	MCC86	ASM86
Z80	XRAYZ80	MCCZ80	ASMZ80
HD64180	XRAY180*	MCC180	ASM180
Gmicro200	XRAYG32	MCCG32	ASMG32
AMD29000	XRAY29K*	MCC29K*	ASM29K*

＊は開発中
● シミュレータバージョンには、コンパイラ、アセンブラが含まれています。
● コンパイラには、アセンブラが含まれています。

プラットホーム環境	
VMS, Ultrix	VAX, MicroVAX
UNIX	HP-9000 Sun-3/4, 東芝 AS-3000 富士通 S-3, 新日鉄 NSSUN ソニー NEWS, apollo
MS-DOS PC-DOS	IBM-PC AT/XT, PS/55 NEC PC-9801, 東芝 J-3100 エプソン PC-286/386, AX

What's XRAYNEWS?

"XRAYNEWS"はC言語シミュレートデバッガ XRAYの販売特約店であるヴイテックから発行しているコミュニケーション誌です。
もともとは、XRAYに関するインフォメーションをユーザへ提供することからスタートした"XRAYNEWS"ですが、現在ではXRAYのユーザだけでなく開発に携わる皆様に対しても愛読していただける小冊子のひとつとなっています。
XRAYのユーザ・レポート、How-to集をはじめ、管理職クラスの皆様の体験レポートやコンピュータ文化のアラカルトなど、読者の生の声をふんだんに取入れ読者同志のコミュニケーションを実現しています。
"XRAYNEWS"は年4回、3月、6月、9月、12月の25日に発行されます。

現在好評発行中のXRAYNEWS Vol.9
- ○RISCかCISCか、昔にもどった? RISC CHIP
- ○XRAYによるデバッグについて
- ○CP/M-68KからMCC68K.ASM68Kへのアプリケーションの移植
- ○京の今昔物語——おいでやす京都へ
- ○XRAY倶楽部の設立

XRAY VAR製品紹介

メモリ640KBを1MBにする『MS-DOSメモリ空間拡張ボード』

MCCでコンパイルできるCのソースサイズを大きく拡張する「MS-DOSメモリ空間拡張ボード」をヴイテックから発売いたします。PC-9801上でこのボードを取り付けることにより、ライン数1800～3000ライン、コードサイズ約120KBまでコンパイル可能となります。ただし、このボードが使用可能なホストマシンは、PC-9801シリーズおよびその互換機です。

★How to XRAY! セミナのお知らせ

- ●日　時：9月21日(木) 午後1:30～5:00
- ●受講料：無料
- ●場　所：(株)東芝虎ノ門ショールーム（J-3100、AS-3000上でのXRAYのデモとコマンド活用法を紹介します。）

※なお、人数に制限がございます。参加ご希望の方はお早めに下記の(株)ヴイテック矢田までご連絡ください。

※DOMAINは米国Apollo Computer社、VMS.Ultrixは米国Digital Equipment社、MS-DOSは米国Microsoft社、PC-DOSは米国International Business Machines社の各登録商標です。UNIXは米国Bell研究所で開発されたOSです。

日本マイクロテック リサーチ株式会社　〒105 東京都港区虎ノ門1-19-9　(TEL)03-502-5598　(FAX)03-502-5589

〈販売特約店〉
●岩崎技研工業株式会社/〒612京都市伏見区下鳥羽平塚町13-3(TEL)075-602-7878 ●株式会社ヴイテック/〒604京都市中京区蛸薬師通烏丸西入ル(TEL)075-211-1256 ●株式会社ABC/〒104中央区築地1-13-10(TEL)03-546-7777 ●エーエスアールインターナショナル株式会社/〒105港区虎ノ門1-19-9(TEL)03-502-5550 ●エーエスアール株式会社/〒105港区虎ノ門1-19-9(TEL)03-502-5571 ●コアデジタル株式会社/〒154東京都世田谷区三軒茶屋1-22-3コアビル(TEL)03-795-5171 ●株式会社ザックス/〒167東京都杉並区荻窪5-20-12(TEL)03-392-3331 ●株式会社ソフィアシステムズ/〒163新宿区西新宿2-4-1新宿NSビル8F(TEL)03-348-7000 ●ダイヤ セミコン システムズ株式会社/〒154世田谷区三軒茶屋1-37-8ワコーレ64(TEL)03-487-0386 ●東京エレクトロン株式会社/〒183府中市住吉町2-30-7(TEL)0423-33-8111 ●株式会社東陽テクニカ/〒103中央区日本橋本石町1-1-2(TEL)03-279-0771

16bit i80186 Board Computer

INP-186

¥113,000 (TTL8MHz)
¥128,000 (TTL10MHz)
¥118,000 (CMOS8MHz)
¥133,000 (CMOS10MHz)
¥185,000 (8087付8MHz)

ターゲットはINP-186
開発はPC98＋MS-DOS
もう普段着な16ビットはこうありたい！

86を使った開発手順

❶PC98拡張ボードを組合わせてシステムを構築する。
❷モニタのコマンドを使用してハードウェアの基本チェックする。
❸C言語/アセンブラを使用してソフトウェアを作成する。
❹コンパイル/アセンブル/リンク(MS-DOS上で動作するソフトと同じ)をし、EXEファイルを作成する。
❺EXEHEX. EXE(付属ソフトウェア)を使用して、H86(HEX)ファイルを作成する。
❻モニタのH86ロードコマンドを使用してINP-186のRAM(又はRAMボード)にダウンロードする。
❼モニタのコマンドを使用してデバッグする。
❽デバッグが完了したら、EXEHEX. EXEを使用して、HEXファイルを作成し、PROMライタでROM化する。
※実際の手順はMS-DOS MAKE. EXE用のファイルが付属ディスケットに記録されています。

(MAKE. EXE用ファイル
INP-186〈モニタROM構築用〉
SAMPLE〈モニタでのダウンロード用〉
SAMPLE 1〈ROM化用〉)

●付属ソースファイル(一部)

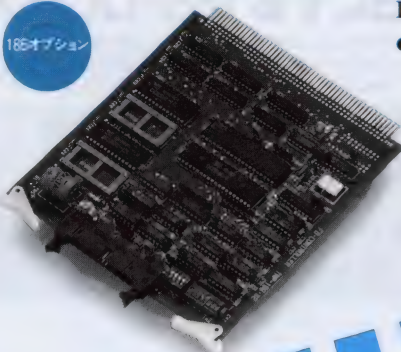
ファイル	サイズ
disasm. c	20812
inp186. c	16620
cbios. c	1622
infr. asm	2101
iolib. asm	4541
bios. asm	14103
inpsub. asm	5099
printf. asm	461
cstart. asm	1506

※ソースはLattice C用に記述されていますが、他のC処理系への移植も容易です。

●EXEHEX機能(ROM化ユーティリティ)

●EXE形のファイルを指定された絶対番地にロケートします。
●出力ファイルは、インテルHEX又はH86フォーマット。
●ブートストラップの自動作成付加ができます。
●ロケーションマップを作成できます。

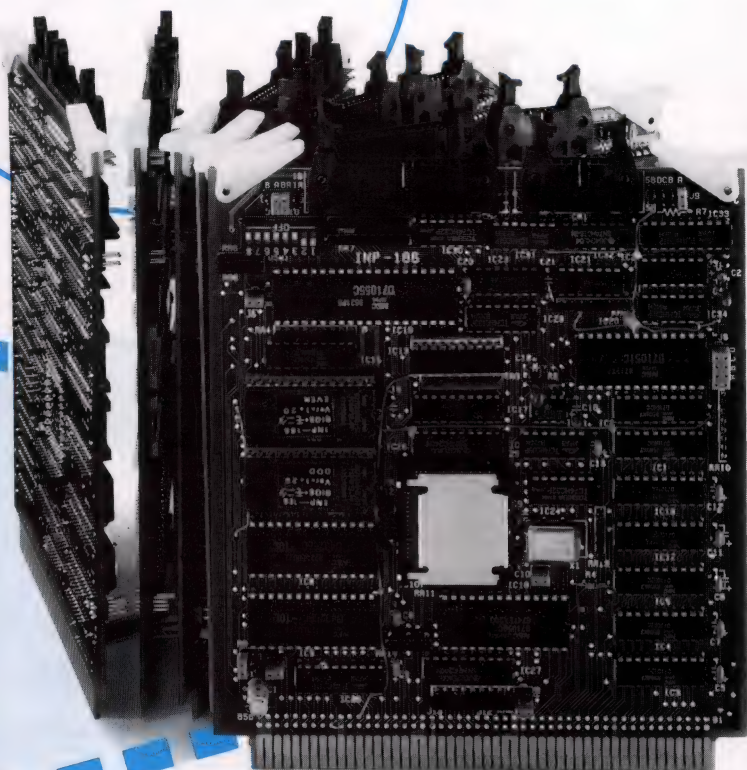
使用可能言語　MASM, Lattice C, Whitesmith C, その他の処理系もスタートアップコードの作成で可能。

●INP-186付属品
回路図/マニュアル/全てのソースファイル
EXEHEX. EXE/ TERM. COM
モニタROM

INP-186用FDインターフェースボード
●INP-186-02……………¥75,000
3.5/5インチFDインターフェースボード
ディスクBIOS組込み済モニタROM付
ディスクBIOSソースファイル付

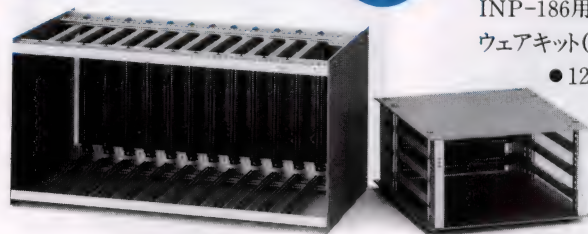
特約店
●九州パーツ株式会社 06-385-4471代　〒564 大阪府吹田市穂波町17番8号
●KHエレクトロニクス株式会社 03-587-1041代　〒107 東京都港区赤坂2丁目10番9号 ランディック第2赤坂ビル

ITO ENGINEERING CO., LTD.

発売以来 大好評！
使いやすさの16ビット。

INP-186を使用したシステム・貴社オリジナルシステムの設計等も承っております。
お気軽に御相談ください。

●INP-186仕様

項目	仕様
CPU	i80186 LCCS (8086' 上位CPU)
CPU clock	8MHz／10MHz ノーウェイト
ROM	64KB 27256×2
RAM	64KB 43256×2(CMOS 完全スタティック)
DMA	2CH (80186内ペリェラル)
割込みライン	10本(80186・8259)外部バス7本、ボード内3本
タイマ	3CH (80186内ペリフェラル)
SIO	1CH 8251 RS-232C 9.6kbps-1.2kbps
PIO	8255×1
バスアービタ	8289／8289-1
バス コネクタ	100P PC-9800シリーズ上位コンパチブル
電源	5V±5% 1.4A ±10% ±12V 0.1A(RS-232C)
外形	約169mm×149mm(PC拡張ボードサイズ)
モニタ コマンド	逆アセンブル、H86ロード、HEXロード、レジスタSET、メモリDUMP・SET・FILL、IN、OUT、GO、トレース
BIOS機能 (ソフトウェアインタラプト30Hで起動)	SIO(標準入出力)、セントロニクス入力に関する12種類(MS-DOS INT21Hの一部に機能的にコンパチブル)

186オプション

- ●EXEHEX……………………¥29,000
  MS-DOS用ROM化ユーティリティ
- ●INP-186-SYS……………¥50,000
  INP-186用MS-DOS作成用ソフトウェアキット(ソース・オブジェクト付)
- ●12スロットカードケージ……
  ¥68,000 19インチラック
  マウンドアダプタ付
- ●INP・186-HDSYS…………¥30,000
  ハードディスク用デバイスドライバ
  (ソース・オブジェクト付)
- ●FD1135C……………………¥36,000
  3.5インチFDドライブ
- ●FD1155C……………………¥32,000
  5インチFDドライブ
- ●3スロットカードケージ ケージ
  組込みタイプ…… ………… ¥23,000

●ご発注の際メディアをご指定ください。
5/2DD・5/2HD・8/2D

●お問い合わせ、資料請求は—— 株式会社
伊藤製作所 〒573 大阪府枚方市宮之下町8-2
0720-53-3553(代) FAX0720-53-3577

SCSIで広がるアプリケーション。

FMR/PanacomM用REX-SCSIシリーズ新登場。

SCSIは、小型コンピュータの周辺機器のための標準バスです。FMR本付内蔵のSCSIポート(簡易型、アービトレイション機能なし)では実現できなかったアプリケーションが、REX-SCSIファミリーで実現できます。たとえば、2台のFMR、2台のラップトップ(M-353、R-50LT)で1台の大容量ハードディスク、光磁気ディスクをSCSIで接続して共有するということも簡単にできます。

注1) REX-6030 SCSIホストアダプタカードはFMR/PanacomMの本体内拡張スロットに実装します。
注2) REX-3530 SCSIホストアダプタカードはPanacomM-353/FM-R50LTの本体内拡張スロットに実装します。
注3) SCSIバス上の大容量ハードディスク、光磁気ディスク上の同一ファイルに対して、4台(上図の例)のパソコンから同時にアクセスすることができますが、ファイルの排他制御機能は、ディスクコントローラ側にありませんのでアプリケーションソフトウェア側で考慮する必要があります。
注4) REX-6030は、FMR/PanacomM本体内蔵のSCSIポートとは全く独立していますので、内蔵ハードディスク、本体のSCSIポートに接続されているハードディスクには全く影響を与えません。

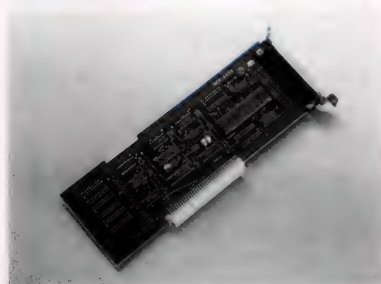
③FMR/PanacomM用ハードディスク

RHD-40 (40MB 3.5インチハードディスクユニット)
SCSIケーブル(1m)1本、ターミネータ付
標準価格**250,000**円(消費税別)

RHD-180S (180MB 3.5インチハードディスクユニット)
SCSIケーブル(1m)1本、ターミネータ付
標準価格**680,000**円(消費税別)

RHD-330S (330MB 5インチハードディスクユニット)
SCSIケーブル(1m)1本、ターミネータ付
標準価格**780,000**円(消費税別)

注1) 上記のハードディスクユニットは、すべて本体標準のSCSIポートに接続して外付ハードディスクユニットとして、そのまま使用できます。
注2) RHD-40は5月より出荷、その他は6~8月出荷予定です。
注3) 使用ドライブは富士通製、または日立製です。
注4) ドライブ、コントローラの基板表面温度が異常に上昇しないよう、匡体、冷却ファン等の配慮を行なっています。

①FMR/PanacomMデスクトップ用 SCSIホストアダプタセット

REX-6030ホストアダプタカード――1枚
SCSIケーブル(1m)――1本
ハードディスク用デバイスドライバソフトウェア――1セット
ハードディスク用ユーティリティソフトウェア――1セット
(FORMAT、DISKPAT)
標準価格**50,000**円(消費税別)

②PanacomM-353/FMR-50LT ラップトップ用 SCSIホストアダプタセット

REX-3530ホストアダプタカード――1枚
SCSIケーブル(1m)――1本
ハードディスク用デバイスドライバソフトウェア――1セット
ハードディスク用ユーティティソフトウェア――1セット
(FORMAT、DISKPAT)
標準価格**50,000**円(消費税別)

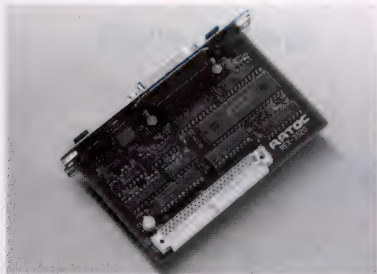
④その他のREX-SCSIファミリー

RMO-333 光磁気ディスク
標準価格**598,000**円(消費税別)
(SCSIケーブル、ターミネータ、ドライバソフトウェア、ブランクディスク1枚付)
ROT-800 追記型光ディスクユニット(発売予定)
RST-220 ストリーマテープユニット(220MB)
標準価格**900,000**円(消費税別)

注5) RST-220は富士通FMMT-102(120MB)の上位コンパチブルです。
注6) 各ユニットに対応したデバイスドライバソフトウェアは別売で提供します。

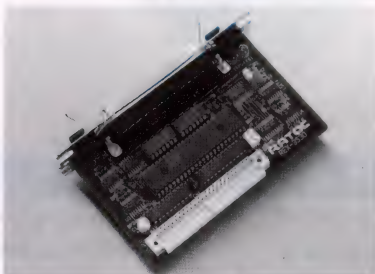
これからは、ラップトップです。

省スペースパソコンには、省スペースI/Fカード。
Panacom M353/FM-R50LT本体内蔵タイプREX-35シリーズ登場。

1 REX-3520M GPIBインターフェイスセット

- GPIBの機能をコンパクトに収めたハードウェア、BASIC、Cから簡単にGPIBをコントロールすることができるソフトウェア、2m長のGPIB機器用ケーブルのセット。
- ソフトウェアは、ディスクトップタイプ(Panacom M、FMR)でのREX-6020M用ハンドラとコンパチブル。BIOSを3520用に入れ換えるだけで、従来のアプリケーションプログラムをそのままラップトップ上で使用することが可能。
- BASIC用リンカは、GPIBバス上の各機器からのSRQ割り込みを、KEY割り込みに変換。SRQ処理ルーチンをON KEY(10) GOSUB＊＊＊の形式で簡単に記述することが可能。

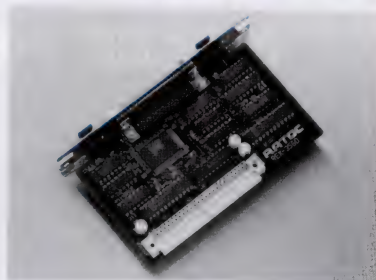
注) 付属ソフトウェア
- 3520用BIOS
- F-BASIC(86)、F-BASIC HG、M-BASIC用リンカ、サンプルプログラム
- Lattice C用関数ライブラリ、サンプルプログラム
- MS-C用関数ライブラリ、サンプルプログラム

標準価格**82,000**円(消費税別)

2 REX-3530 SCSIホストアダプタセット

- ハードディスク、光磁気ディスクドライブなどのSCSIバス周辺機器を接続するためのSCSIホストアダプタカードと、MS-DOS用デバイスハンドラ、ユーティリティソフトウェア、1m長のSCSIケーブルのセット。
- アービトレイション機能など、SCSIの標準機能をサポート。他のパソコンとSCSI周辺機器(大容量ハードディスク、光磁気ディスクなど)を共有することができる。

注) 付属のデバイスハンドラソフトウェア、ユーティリティはハードディスク用です。

標準価格**50,000**円(消費税別)

3 REX-3560 RS232Cレベル、1chの増設シリアルインターフェイスカード

- RS232Cレベルの1ch増設シリアルカード。M BASIC、FBHGのCOM1:またはCOM:3として、またはMS-DOSのAUX1、AUX3としてそのまま使用可能。
- CI(コールインジケータ)、CD(キャリアディテクト)を持ち、モデムとの接続も可能。

(注:非同期モデムに限ります。)

標準価格**40,000**円(消費税別)

4 REX-3562 RS422、RS485レベル1chの増設シリアルインターフェースカード

- RS485レベルを有する機器との間でのマルチポイント通信が可能です。BASIC、Cでマルチポイント通信を可能にするためのソフトウェアを3.5″フロッピーディスクで供給。
- RS422レベルで、1対1の通信を行なう場合は、BASICのCOM1、COM3。MS-DOSのAUX1、AUX3としてそのまま使用可能。ボーレートはBASICのBAUD文または、MS-DOSのセットアップユーティリティで300〜19,200bpsより選択。RS422、485の選択は基板上のスイッチで選択。
- RS422、RS485非同期シリアルポートを持つNC工作機器との通信、RS232Cに比べて高速、長距離、耐ノイズが要求されるアプリケーションに使用。

標準価格**42,000**円(消費税別)

RATOC

ラトックシステム株式会社
〒556 大阪市浪速区敷津東1-6-14朝日なんばビル5F
TEL.06(633)8263㈹ FAX.06(633)8295

〈販売代理店〉
- 富士通ビジネスシステム全国各営業所
- ナショナル電子計測全国各社
- ナショナル通信特機全国各社
- 関東電子全国各営業所 他

資料請求No.94

パーソナルコンピュータとともに

EAST

C言語ソースプログラム集

C4pro PostScriptライブラリ

¥320,000(マニュアルのみ¥3,000)

●NEC PC-PR201系のエスケープシーケンスに準拠した体系を持つC言語関数ライブラリ集で、PostScriptコマンドを出力します。
●MS-DOS上で稼働しているアプリケーションソフトをPostScript対応に変更するためのライブラリです。
●PC-PR602のグラフィックコマンド互換の関数ライブラリも備えています。
●このライブラリのサンプルプログラムとしてフォーム出力ユーティリティも提供します。
●proシリーズはライセンスフリーですので、このライブラリを組み込んだソフトは、自由に販売出来ます。
●日本語PostScriptの勉強にも最適です。
●250頁のマニュアル付き。

PC-PR602PS 対応

【製品内容】

●**テキスト系ライブラリ**(48種)
印刷指令、改頁、漢字書体選択、スクリプト文字制御、組文字、文字サイズ設定、文字幅指定、ドットスペース指定、改行幅指定、拡大率設定、字飾り(強調、イタリック、アウトライン、網かけ)

●**グラフィック系ライブラリ**(27種)
スケーリング、回転角、線種設定、ペン移動(アップ、ダウン、絶対描画、相対描画)、円描画、円弧描画、扇形描画、楕円描画、塗潰し(枠あり、枠なし、閉領域、パターン登録、モード設定)

●**サンプル・プログラム**
使用方法を解説した各種のサンプルプログラムが入っています。

C5pro TIFFライブラリ

¥320,000(マニュアルのみ¥3,000)

●イメージファイルの標準化として各分野で注目を集めているTIFF(タグイメージファイルフォーマット)のC言語関数ライブラリです。
●TIFFファイルのopen/close/read/writeの他、多数のイメージ処理ライブラリが入っています。
●イメージ圧縮はCCITT/3の1次元圧縮、GⅢ、GⅣの3種類をサポートしており、このロジックもソース公開しています。
●proシリーズはライセンスフリーですので、このライブラリを組み込んだソフトは、自由に販売出来ます。
●TIFF形式、イメージ処理の勉強にも最適です。
●400頁のTIFF仕様を含む詳細なマニュアル付き。

【製品内容】

●**TIFFライブラリ**(24種)
►ヘッダ関係…TIFFオープン、新規作成、ヘッダ情報のread/write/set
►ディレトリ関係…TAG情報のread/write、TAG情報昇順設定
►TAG関係…TAG数の取得、TAG構造のチェック、TAGread、解像度変更、RGBモード変換、ストリップ切り分け
►その他…イメージ圧縮/伸長、インテル/モトローラ変換、ビットイメージ/TIFF変換

●**サンプルプログラム**
TIFFライブラリの上位ルーチンとしてTIFFファイル全体に対する操作を行なう。

●**TIFFファイル操作ユーティリティ**
►TIFFダンプ…TIFFファイルの形式を画面表示する。
►DataSetter…会話型ビットイメージ、TIFF変換プログラム。

C1 かな漢字変換ライブラリ

¥80,000

●最新の2文節最長一致法を使用していますので「彼は医者で彼らは歯医者です」「代表取締役」などもイッパツで変換します。
●以下のようなC言語プログラムインターフェースを関数として用意しています。
►ひらがな文字列を漢字混じり文字列に変換する
►次の候補文字列を得る
►ローマ字をひらがなに変換する　など
●複合連文節変換/活用語尾変換処理/付属語処理/数字変換/辞書更新学習機能などの最新ロジックも入っています。
●25,000語の厳選された辞書ファイル付き。

C2 汎用スクリーンエディタ

¥80,000

●ANSI標準エスケープシーケンスを使っていますので各社のMS-DOSパソコンで稼働します。
●Read/Write/Jump/Cut/Paste/Find/Substituteなどの基本機能のほか、WordMaster系のCTRLコマンドも持っています。
●スクリーンエディタとしての基本用件に加え、以下の機能を持っています。
►仮想テキストバッファ処理(メモリとファイルのバッファリング)
►日本語編集(半角/全角を考慮したカーソル移動、各種カナ漢FEPとの併用)
►OSコマンド実行(編集中にMS-DOSコマンドを実行可能)
►階層構造ディレクトリのサポート

■C言語ソースプログラム集販売条件■

【C1、C2】●提供する製品(プログラム、仕様書、データなど)は、購入者個人が調査、研究する目的のものです。第三者へのコピー提供、開示、再販売は行えません。
●製品の全部もしくは一部を、転用、転載、改造することを禁じます。
【C4、C5】●製品の全部もしくは一部を、ソースプログラムのまま転用、転載、開示、販売することを禁じます。
●実行形式での販売は自由に行なえます。
【共　通】●詳細なコメント、仕様書付きですので、仕様についてのお問い合わせにはお答えできません。
●本製品は、調査、研究を目的とした販売ですので、運用した結果を保証するものではありません。運用した結果については、責任を負いかねます。

C言語ソースプログラム集とは……

●ANSI規格に準じたC言語で記述していますので以下のようなコンパイラでコンパイルできます。
►MicroSoft C　V3.0以降　►Turbo C　V1.5j以降
►Lattice C　V3.0以降　►Quick C　V1.1以降
他のコンパイラについては、稼働確認は行なっておりませんがANSI準拠であればコンパイル可能です。
●製品内容は以下の通り
►フロッピィディスク(MS-DOSフォーマットの5"2HD)
・ソースプログラム　.C　(日本語ヘッダコメント付き)
・実行形式プログラム　.EXE
・サンプルプログラム　.C　.EXE　(ソースと実行形式)
・link/makeパラメータファイル
►プログラム仕様書(50~400頁)
・機能仕様、モジュール関連図/共通バッファ仕様
・関数仕様、サンプロプログラムの操作方法などについて記載
●各社のMS-DOSパソコンで稼働します。ソース提供ですので、技術力しだいではUNIX OS/2プレゼンテーションマネージャーなどへの移植も可能です。

MS-Windows OS/2バージョン1.1(PM)

EISAM データベースライブラリ

¥128,000

●業務アプリケーションを開発する際に必要となる、検索、並べ換え(ソート)などの基本操作を行なうライブラリです。
●B+tree方式により、大容量のデータ検索も瞬時に行なえます。
●パソコンメーカー2社へのOEM実績および3年間の販売実績を持っていますので、高速かつ安定した動作を保証します。
●常駐型とライブラリ型の2タイプで提供しますので、MS-DOS上の各種言語から利用可能です。
●インデックス圧縮/4バイト整数、浮動小数点のサポート/重複キー/レコード数無制限/電源断対応など1クラス上の機能を用意しました。
●各言語でのプログラミング例がマニュアルおよびサンプルファイルとして付いています。
●サポート言語
►MicroSoft C (S/M/L) ►Lattice C (S/P/D/L) ►Level2 COBOL
►MS-BASIC インタプリタ/コンパイラ ►C86-BASIC
►N88-BASIC インタプリタ/コンパイラ
●PC-9800シリーズ、各社AXパソコン、IBM55シリーズなどの各社MS-DOSパソコンに対応しています。
●業務アプリケーションへの組込みロイヤリティ ¥8,000/本

かな漢字変換

E1 日本語入力フロントプロセッサ

¥18,000

●E1は、MS-DOS版とMS-Windows版の2本のプログラムがいっしょにパッケージされています。

	DOS版	Win版
稼働機種	PC-9801シリーズ (LTおよび高解像モードを除く) AXパソコン	日本語MS-Windows 2.0以降が稼働するパソコン
必要メモリ	76KB	55KB
サポートソフト	MIFES、final、RED、TurboC、QuickC、QuickBASICなど	ExcelなどMS-Winで稼働する全アプリケーションソフト

●MS-KANJI APIに準拠していますので、 マイクロソフト社のクイックシリーズ(QuickC、QuickBASIC)などで心地よい操作が行なえます。
●占有メモリ76KB以内(辞書バッファを含む)と他のFEPの約半分のメモリしか使いません。(Win版)
●辞書はひらがな、外来語を含め40,000語以上入っていますので、確度の高い変換が可能です。※αβR①Ⅷ㎡㈱など、すべての特殊文字に読みが付いています。
●変換文字色のユーザ設定、再変換機能による郵便番号変換、半角変更による数字の無条件半角入力、無変換キーによるカタカナ変換、XFERキー単独でFEPのon/offなどかゆいところに手が届く仕様を詰め込みました。
●30分で操作が覚えられる、トレーニングテキストファイル付き。
●MS-DOS、MS-Windows、OS/2のOEM商談受付中。

OEM用 日本語入力ソフトウェア

¥3,000,000〜

●Z80アセンブラ版(2種)、8086アセンブラ版(2種)、C言語版 (MS-DOS用、OS-9用、68000用)など多くのソース資産を用意しています。
●辞書インタフェースライブラリ(C1)のソース販売から、かな漢字変換操作部分を含むフルパッケージの販売そして移植作業まで、各種形態に対応致します。

MS-Windows、OS/2関連

●MS-Windows、OS/2プレゼンテーションマネージャーについて、以下のようなソフトウェアの企画/開発を行なっております。作業依頼、OEM、パッケージ販売など、各種形態に対応します。以下のイーストプロダクトを使った業務アプリケーションの受注開発もお受けしております。

日本語ワードプロセッサ(TextWriter)

●マルチフォント(明朝体、ゴシック体)、自由な大きさの文字(5〜63ポイント)をサポートした、MS-Windows対応の日本語ワープロです。
●豊富な図形罫線機能により斜め線を含む、表現力豊かな文章が作成できます。TIFF(イメージ)の読み込みも可能です。
●日本語Excelのデータを取り込みレポート出力が可能です。
●日本語PostScriptプリンタ、NECマルチフォントカード対応。
●文書ファイルは、拡張テキストファイル(ETX)を使用しており、各種用途に柔軟に対応します。

データベースソフト(DataFolder)

●画面定義体、ISAMインターフェース(OS/2では、SQLインタフェース)を持つデータベースシステムです。
●カラーイメージ付きのファイルやTextWriterの文書ファイルの検索システムから、業務処理プログラムのマスターファイルメンテナンスや顧客管理システムなど幅広い用途で使えます。
●SQLサーバーを使ったローカルネットワークシステムにも対応します。

表計算・グラフソフト(WinCalc/WinGraph)

●レポートライター機能付の簡易表計算プログラムです。
●Excelでは機能が大きすぎる場合や帳票出力時の簡単なシミュレーションなどにお使いください。

日本語入力ドライバー(E1Win)

●MS-Windowsのアプリケーションインタフェースに対応した日本語入力ドライバーです。KKAPP型とMS-Winのアプリケーション型の2種類を用意しました。

イメージ処理ソフト(ImageEditor)

●イメージスキャナから読み込んだ画像情報を画面に表示し、 編集後、イメージファイルの共通フォーマットであるTIFFファイルを生成するプログラムです。
●イメージの圧縮、カラー、ディザなどをサポートします。

フォント編集ソフト(FontEditor)

●任意の大きさの漢字ビットマップフォントおよび漢字ベクターフォントの編集を行なうプログラムです。
●ベクターフォントは、PostScript形式のEPSファイルで蓄積されます。
●ビットマップ→ベクター、ベクター→ビットマップのデータ変換が可能です。

プリンタドライバー

●シリアルプリンタ、ページプリンタ、PostScriptプリンタ用の各種プリンタドライバーについて、豊富な開発実績を持っています。

HeartBeatDemoPack(¥8,000)

●MS-Windows 関連のデモソフト、 デモデータを販売しています。 TexWriter、DataFolder、E1Win、TIFF関連ユーティリティ、カラーアニメーションなどのデモソフトが入っています。郵便振替またはFAX注文でお申込み下さい。

※ご注文は郵便振替でお受けしております。
東京2-62623 名義:イースト株式会社

※TIFFはマイクロソフト社とアルダス社が考案したイメージファイルフォーマットです。※PostScriptはアドビ社の商標です。※MS-DOS、MS-Windows、MS-C、Quick C、Excelはマイクロソフト社の商標です。※Turbo Cはボーランド社の商標です。※Lattice Cはラティス社の商標です。※UNIXはATTが開発したOSです。

イースト株式会社

〒151 東京都渋谷区代々木1-3-1 TEL03(374)1980 FAX03(374)2998

SOFTWARE Library

Number 55

■和文マニュアル付 □英文マニュアル付
◪和文又は英文マニュアル付(機種対応のものあり)
ⒸCP/M-86 ⓂMS-DOS ⓅPC-DOS
ⓀCP/M68K

16Bit Software

OrCAD シリーズ

■ **OrCAD/SDTⅢ** - 階層化設計やラバーバンディング機能、3,700種類のライブラリ、ライブラリパーツの追加/登録、自動割付機能、エラーチェック、パーツリスト、22種類をカバーしたネットリストを標準装備させたパーソナル CAD ……………… ⓂⓅ**¥168,000**

■ **OrCAD/VST** - 12ステートまで扱えるイベントドリブン論理シミュレータ。毎秒10,000イベントで14,000ゲートまでの処理が可能。ブレークポイントは10個までの設定が可能 ……………… ⓂⓅ**¥298,000**

■ **OrCAD/PCB** - 16レイヤー、2 シルクスクリーン、2 ソルダーレイヤー、32インチ×32インチまでの基板作成が可能。2,000ネット、1,850パッド、130ICまで取扱可能。強力なオートルーティング機能をサポート。……………… ⓂⓅ**¥598,000**

■ **OrCAD/PLD** - OrCAD/PLD は、SDTⅢで作成されたロジック回路を論理圧縮し、今までのコンパイラでは真似のできない、高速でオプティマイズ化された PLD 設計が行なえます。……………… ⓂⓅ**¥148,000**

■ **OrCAD/MOD** - OrCAD/MOD は、PLD 情報を VST のライブラリファイルに変換し、VST 上で PLD のシミュレーションが行なえます。……………… ⓂⓅ**¥148,000**

■ **ALS-VIEW** - ALS-VIEW は OrCAD/PCB をはじめ、いろいろな PCB CAD で出力されるGERBER フォーマットのファイルを高度にエディトする事が出来ます。……………… ⓂⓅ**¥398,000**

Computer Innovation シリーズ

■ **C86plus C Compiler** - C ライブラリ関数がソースコード又は、オブジェクトコードに変換可能な為、関数の変更やライブラリファイルへの追加が可能。UNIX SYSTEM V. ANSI C の最新規格を取り入れたフルセットの ANSI C コンパイラ。従来の OPTIMIZING C86、及び MS-C のデータは C86 plus でも使用可能。機能アップされたコンパイラーと専用リンカを使用する事により OPTIMIZING C86(V2.30)を70%、MS-C(V4.0)を20%アップさせたコードを生成。ANSI C は、レジスタ変数、長倍精度80ビット浮動小数点などをサポート。コンパイラは、スモール、ミディアム、ラージの3モデル、CPU は8086～80286、8087/80287のインラインフローティングポイント、エミュレータ、デバッガユーティリティ、ワイルドカードコンパイル、MAKE ユーティリティ、ROM 化ユーティリティ、マクロアセンブラ出力などのサポート。ライブラリは、250以上の関数、ANSI C フルセット、UNIX SYSTEM V と同等な関数、IBM ROM BIOS、などをサポート Ver1.2(OEM 御相談下さい) ……………… Ⓜ**¥98,000**

□ **ROMpacPLUS** - C86plus専用ROM化ツール、C86plusで作成されたロードモジュールをインテルHEX フォーマットにトランスレート、大幅機能アップバージョン… Ⓜ**¥98,000**

□ **SOFTPROBEⅡ/TX** - インテル8086/87/186/286用ソースレベルデバッガ IBM PC/XT/AT 及びそのコンパチマシン用のみ ……………… **¥59,000**

□ **C86PLUS/ROM C COMPILER**-C86PLUS C と ROMpacPLUS のセット高度にオプチマイズ化された小さくて高速なコードの ROM 化開発が可能。データエリアからコードエリアへリードオンリデータの移動化、インテルスタイルのリンカ/ロケータやデバッギングがサポート……………… Ⓜ**¥189,000**

□ **C QNX** - マルチタスク・マルチユーザをサポートしたリアルタイムオペレーティングシステム用 C コンパイラ。

- Single Node …… **¥150,000** ● 4 Nodes …… **¥300,000** ● 8 Nodes …… **¥450,000**
- 16Nodes …… **¥600,000** ● 24Nodes …… **¥750,000** ● 32Nodes …… **¥900,000**
- 40Nodes …… **¥1,200,000** ● Manual …… **¥20,000**

Phoenix 開発ツール

□ **Plink86Plus** ●インプットされるオブジェクトを 2 回ロードし、初めはメモリ内にモジュール、セグメント、テーブルを設定し、最後にこれらの情報を基にオブジェクトをリンクする事によりシステムのメモリ限度よりもはるかに大きなオーバーレイされたロードモジュールの生成が可能●スタティックシンボルをパブリックシンボルに指定したり、Group、Class、Module の中のセグメントをカレントセクションに割当てることが可能●大規模なプログラムでメモリが足りない場合や、データ領域を多く必要とする場合、プログラムをあるセクションで分割し、その機能が必要になった時点で補助メモリ(ハードディスクやフロッピーディスク)からメインメモリにロードして実行することが可能● RELOAD ステートメントによる同一オーバーレイエリア内での呼び出しも、自動的にそのセクションの再ロードが可能●ファイルの中で必要でない関数を除外してライブラリに追加するといった操作が PLIB86により可能 ……………… Ⓜ**¥98,000**

□ **Pfix86plus** ● Pfix86plus は、多種のコンパイラに対応し、動作メニュー、マルチウィンドをサポートしたダイナミックシンボリックデバッガ● Plink86plus や MS-LINK で生成されたシンボルファイルや Plink86plus で生成されたオーバーレイファイルなどでもダイナミックにデバッグが可能●ソースコード、アセンブラインストラクション、スタック、データエリア、ブレークポイントなどを同一画面上に表示することが可能●一時的なパッチをインラインアセンブラで行なうことや一時的なブレークポイント、持続的なブレークポイントの設定が可能●高速トレースモード、変数のユーザ指定、100ステップまでのトレースバック、ディスク又はプリンタへのデバッグ記録、ソースコード上のブレークポイント設定、ディスクへの逆アセンブラ記録、メニューのコンフィグレーション、キーストロークのマクロが可能。*値下げ* ……………… Ⓟ**¥78,000**

□ **Pmate** ● Pmate はフルカスタマイズが可能な高機能テキストエディタ●バックグラウンドでの実行能力、C や Fortran の特殊マクロ、自動バッファリング、10個の補助バッファ、水平スクロール、広範囲なマクロコマンドによるメニュー、マウス(IBM-PC のみ)、コマンドの動作などをサポート●オートマチックワードラップ、テキストフォーマット、グローバルおよびローカルのセッティング、タブストップ、インデント、消去した項目を回復させセーブするユニークなガーベッジスタックなど可能 ……………… ⓂⒸ**¥47,000**

□ **PforCe** ● PforCe は400種類以上もの最適化されたオブジェクトを集めた C プログラマーへのツールキット● DATA BASE 開発に必要な、B-Tree、Windows ライブラリのサポート●割込み動作による通信ライブラリ、メニューライブラリ、全てのベーシック DOS インターフェースライブラリ、C コンパイラによる全てのメモリモデルをサポート●ライブラリソースコードでユーザが作成したアプリケーションソフトウェアについては、ノーロイヤリティ… Ⓟ**¥95,000**

□ **Pasm86** ● 8086～286、8087～287までをサポートし MASM4.0より高速な MASM コンパチブルアセンブラ ……………… Ⓜ**¥47,000**

□ **Pdisk** ●バックアップ/リスト、メニュー、ファイル操作を含むディスク管理パッケージ。これらにより MS-DOS 使用の簡易化、動作のスピードアップが可能。*値下げ* …… Ⓟ**¥20,000**

□ **Pmaker** ● UNIX の MAKE 機能を MS-DOS 上でサポート……………… Ⓟ**¥30,000**

□ **Pfinish** ●アセンブラ、C、PASACL、FORTRAN、BASIC、のプログラム実行速度を最大出力レベルに高めるツール。リンカが出力するシンボル情報を利用し、プログラムの不効率な部分をスキャンし、サブルーチンや関数の使用頻度を表示する。*値下げ* …… Ⓟ**¥48,000**

AVOCET 開発支援ツール

□ **AVXC シリーズ** - ランタイムソースコード付 ROM 化用 C コンパイラ。浮動小数点演算、UNIX ライク関数、重要な ANSI C 関数などをサポート。……………… Ⓜ**¥198,000**

- **AVXC11** - 68HC11Family ● **AVXC68** - 6800Family
- **AVXC51** - 8051Family ● **AVXCZ80** - Z80Family

□ **AVMAC シリーズ** - マクロ機能を持つリロケータブルクロスアセンブラ、リンカクロスリファレンス、ライブラリアン、エラーメッセージ機能付。ターゲット CPU は以下の如くラインアップされ広範囲な開発環境を提供する。英文マニュアル及び和文サマリー付 … Ⓜ各**¥75,000**

- **AVMAC04**-6804 Family
- **AVMAC05**- All 6805、146805、68705、1468705、68HC05、HITACHI 6305
- **AVMAC09**- All 6809 Family
- **AVMAC68**-6800、6801、6802、6803、HITACHI 6301
- **AVMAC11**-68HC11Family
- **AVMAC48**-8021、8022、8035、8039、8048、8049、8050、8041、8041A、etc.
- **AVMAC51**-8031、8032、8044、8051、8052、8751、etc.
- **AVMAC96**-8096 Family
- **AVMACZ8**- Z8 Family
- **AVMAC18**- RCA1802、1804、1805、1806
- **AVMAC65**-8bit 6500 family: 6500、6502、6511、65C02、etc.
- **AVMAC85**-8080、8085、Z80 (Intel Mnemonics)
- **AVMACZ80**-64180、Z80、8080、8085 (Zilog Mnemonics)
- **AVMAC321**- TMS 32010、32015
- **AVMAC322**- TMS 32020、320C25

□ **XASM シリーズ** -各 CPU のニーモニックで作られたプログラムをインテル HEX(モトローラ S)フォーマットに変換したファイルとプリントファイルを生成。更に多数の擬似命令をサポート。次の各種が揃っている。

- XASM -04 ● XASM -05 ● XASM -Z8 ● XASM -09 ● XASM -18 ● XASM -48
- XASM -51 ● XASM -65 ● XASM -68 ● XASM -75 ● XASM -85 ● XASM -Z80
- XASM -F8 ● XASM -400 ● XASM -180 ● XASM -11 ……………… Ⓒ各 **¥67,500**

□ **XMAC68K30** -従来の XMAC 68 K(68000/10)及び68K20(68020)に加えて68030、68882まで サポートする新バージョン。アセンブラインストラクションは、モトローラ68000/10/20/30に準拠。その他拡張命令をサポート。マクロ機能やインクルードが可能なプリプロセッサ付。最適化されたアセンブリと構造化アセンブリが可能。リンカはインテル HEX、モトローラ、モステックスのオブジェクトフォーマットが可能でありオブジェクトサイズはホストメモリーに依存し、特に制限はなく、クロスリファレンス、メモリーマップ、シンボルなどの出力も可能。シンボルテーブル、クロスリファレンス、メモリーマップなどの作成は、シンボル・レポート・ジェネレータが行ない、ライブラリ管理、リストモジュールの追加、削除はライブラリアンで行なう。その他、モトローラオブジェクトファイルをインテル8ビット外部バス用オブジェクトファイルに変換する SPLIT 機能。モトローラオブジェクトファイルをバイナリファイルに変更可能な IMAGE 機能をサポート。……………… Ⓜ**¥198,000**

□ **AVSIM シリーズ** -シュミレートをしながらのフルスクリーンデバッグが可能。ジャンプ先、スタックエラー、割り込み処理もサポート、ターゲット CPU のフラグ、レジスタ、スタック、マシンコード、データメモリエリア、インストラクションサイクルなどの内容がスクリーンにウインドとして表示。各 CPU の詳細(ROM、RAM、マスクオプション、インストラクションセットタイミング)が区別でき、ROM への書き込み、存在しないメモリーの参照、コントロールレジスタ、I/O ポート、タイマなどのエラーをキャッチ。オペレーションは、プロンプト、メニュー、ヘルプコマンドにより操作は簡単。各機能はファンクションキーをサポート。デバッグ機能は、シングルステップ、ブレークポイント、プレイバック機能をサポート。ターゲット CPU は以下の如くラインアップ ……………… ⓅⓂ各**¥120,000**

- **AVSIM05**-6805 : P2、P4、P6、R2、R3、U2、U3、68705 : P3、P5、R3、R5、U3、U5、146805 : E2、E3、F2、G2、1468705 : G2
- **AVSIM09**-6809 family, with optional 6821 PIA and/or 6840 PTM and/or 6850 ACIA

● AVSIM11-68 HC11A8、68HC11A0、68HC811A2、with optional 68HC24PRU
● AVSIM48-8020、8021、8022、8035、8039、8040、8041、8041A、8042、8048、8049、8050、8641A、8741A、8742、8748、8749、80C35、80C39、80C48、80C49
● AVSIM51-8031、8032、88051、8052、8751、8752、80C31、80C51
● AVSIM65-6502、65C02、with optional 6522 VIA and/or 6551ACIA
● AVSIM68-6800/6802/6808、6801/6803、6801U4/6803U4、6301V1/6303V1、with optional 6821 PIA and/or 6850 ACIA
● AVSIM85-8085 CPU、with optional 8155 and/or 8355
● AVSIM Z80- Z80 CPU、with optional CTC and/or PIO
● AVSIM321- TMS 32010、320C10、320C15、320E15
● AVSIM322- TMS 32020、320C25
● AVSIM180-64180
● AVSIM96-開発中
● AVSIM68K -開発中

□ PASCAL クロスコンパイラシリーズ- Ⅰ-codeによりアセンブラソース出力可能。アセンブラソースはターゲット CPU の他8086(スモール)コードをサポート、AVMACと組み合せることにより ROM 化可能、以下のターゲット用ラインアップ ……… Ⓜ各¥89,000
● MXPASCAL-51 ● MXPASCAL Z80 ● MXPASCAL 68K

□ MXGEN シリーズ- PASCAL コードを最適化されたアセンブラコードに変換する。以下のターゲット用ラインアップ。……… Ⓜ各¥38,000
● MXGEN 51 ● MXGEN Z80 ● MXGEN 68K

System Tools

■ Super Lattice(サピエンス)- EWSに優るとも劣らないスピードを実現した PC-9801専用ハイレゾリューション・ディスプレイ・インターフェース……… Ⓜ¥178,000
■ OBJ2HEX(コントロールプロセス)-マイクロソフトフォーマットのオブジェクトライブラリをリンクし、インテル HEX フォーマットのファイルが生成可能 ……… Ⓜ¥98,000
■ ROM-GEN2(シンフォニー)- MASM で ROM 化を可能にするツール、MS-LINK で出力された EXE ファイルからセグメントをリロケートし、インテル HEX ファイルを生成する ……… Ⓜ¥38,000
□ VEDIT plus (Compu View) -あらゆるパソコンに対応した、最強のフルスクリーンエディタ。37個のファイルを同時にオープン、DOS コマンド Shell、電卓機能、オートマチックインデント、カッコチェック、IF、THEN、ELSE、ループ、テスト、キー入力、変数、ファイルのコンペア、ソート、マージ、などの機能をサポート ……… Ⓜ¥49,800
■ WordMaster86(マイクロプロ)-フルスクリーンエディタ……… ⒸⓂ¥38,000
■ WordMasterPlus(マイクロプロ)- WordMaster + VJE α ……… Ⓜ¥45,000
■ VX(ハートコンピュータ)スクリーンエディタ……… ⓀⒸⓂ¥38,000
■ FINAL (SPS) -究極の漢字スクリーンエディタ Ver4.0フリーカーソル、正規表現、ビルトイン grep、ヒストリー、カスタマイズ機能、カット&ペーストなどの機能をサポート ……… Ⓜ¥38,000、VJE β 付¥48,000、PC98XA/XL 用¥48,000
■ Z/TRN(大信機器)- Z80及び8080アセンブラソースに最小の変更を加える事によりASM 86用のソースファイルを生成する……… ⒸⓂ¥38,000
■ HGD-1(シンフォニー)-日立 B16用グラフィクライブラリ ……… Ⓜ¥58,000
■ PLMTOC (ADC) - PLM86から C へのトランスレータ……… ⒸⓂⓀ¥300,000
□ BASTOC (ADC) - BASIC ソースから C ソースへのトランスレータ……… Ⓜ¥150,000
■ The-BC (CRC システム)- N88BASIC から MS-C へのトランスレータ……… PC-9810用Ⓜ¥88,000

C Langages

■ MS-C Compiler(マイクロソフト)-ウインドウ指向のデバッガ「CODEVIEW」、Ver5.1 ……… Ⓜ¥98,000
● SYM-232C - PC-9861K インターフェース対応、ソースコード付(for MS/Quick C)。……… Ⓜ¥50,000
● SYM-232C - PC-9861K インターフェース対応(for Quick C)。……… Ⓜ¥28,000
● SYM-BGM - C 言語用グラフ作成ライブラリ。円/棒グラフなど12種類のグラフ作成関数集。ソースコード付、ランタイムライセンス料なし(for MS-C)。……… Ⓜ¥98,000
● SYM-RDB - C 言語用データベース作成ライブラリ。画面設計/帳表設計/ISAM 関数集(for MS-C)。……… Ⓜ¥68,000
● SYM-RDB - C 言語用データベース作成ライブラリ。画面設計/帳表設計/ISAM 関数集(for Quick C)。……… Ⓜ¥68,000
● MS-C(Ver5.1)+ C 拡張関数ライブラリ - MS-C にBASIC ライクに使用できる関数集をバンドル。……… Ⓜ¥99,800
● RX-GRAPH - 沖 RX-110用グラフィックライブラリ(for MS-C)。……… Ⓜ¥28,000
● SYM-HILIB - MS-C 用高速化、コンパクト化ライブラリ(シングルユース) ……… Ⓜ¥29,800
組み込み権付バージョン(プロユース)。……… Ⓜ¥148,000
● Quick C +「入門ビデオ」- マイクロソフトの Quick C に「入門ビデオ」をシンフォニーがバンドル。……… ¥23,800
● SYM-WINDOW - MS-C 用ウインドウライブラリ ……… FMR50用Ⓜ¥48,000 ……… FMR60用Ⓜ¥50,000
● SYM-GRAPHⅢ - MS-C 用グラフィクライブラリ……… FMR50用Ⓜ¥38,000
● GP-IB ライブラリー ……… PC-9801用Ⓜ¥75,000
■ Lattice C Compiler(ラティス) ……… Ⓜ¥98,000
オプションシリーズ(98用のみ) ……… 各¥29,800
C-TOOL/98、C-PROFILER/98、C-SPRITE、C-ISAM、C-RUNTIME、C-SSP、C-MASM、C-ENTRY/98、C-WRITER/98、C-日本語/98、C-GPIB/98、C-WINDOW/98、C-MATH/98、C-MODEM/98、C-LOCAT

● LINK & LOCATE (SSI)……… Ⓜ¥158,000
■ Hi-TECH C(ハイテック)……… ⒸⓂ¥47,500
□ Whitesmiths' C Compiler ……… 各種 OS サポート
● C ネイティブコンパイラ -UNIX Ver7・C ランゲージに相当、システムウェアツールの極限。C ソースプログラムをアセンブリ言語へコンパイル、C から各 CPU へのクロス開発に便利、Ver3.0はラージモデル完全サポート、リスト機能強化、アセンブラと C とのリストの混合OK。C ソースレベルでの会話型シンボリックデバッガ装備。
● クロスサポート ● C クロスコンパイラ ● PASCAL サポート
● PASCAL ネイティブコンパイラ ● PASCAL クロスコンパイラ

FORTRAN Langages

■ MS-FORTRAN.(マイクロソフト)- Ver4.0 ANS177規格に、完全準拠したヒュージモデル、8087/80287、他言語とのリンクをサポート、ウィンド指向のデバッグ「CODE・VIEW」付 ……… Ⓜ¥80,000
■ Pro Pack 98 ……… Ⓜ¥168,000
■ Pro FORTRAN-77(プロスペロ) ……… Ⓜ¥98,000
■ FORTRAN 77コンパイラ (absoft) ……… Ⓚ¥300,000

BASIC Langages

■ TKW-86BC(西日本常磐商行)-高速演算処理用の FORTRAN ライクな BASIC コンパイラで科学技術計算及び計測分野に最適。PC 9801用のみグラフィック付 ……… ⒸⓂ¥45,000
■ MS-BASIC(マイクロソフト)-3レベルでのサポート(8K、拡張 BASIC、ディスク BASIC)ANSI 標準 ……… Ⓜ¥48,000
■ MS-BASCOM(マイクロソフト)-オブジェクトコードはリロケータブル、マシンコード出力可、倍精度関数、% INCLUDE、CHAIN、COMMON 文サポート ……… Ⓜ¥48,000
● GP-IB ライブラリー - Quick Basic 用 *NEW* ……… Ⓜ¥75,000

その他の Langages

■ Pro PASCAL(プロスペロ)…PC9800シリーズ対応 ……… ¥98,000
■ TURBO-PASCAL(ボーランド) ……… Ⓜ¥29,800
■ Fifth-86(リギー)-構造化プログラミングの極致、アセンブラ的融通性、自己増殖性、対話型デバッグ機能あり、浮動小数点サポート ……… ⓀⒸⓂ¥90,000

Utilites & Applications

■ JF-TRAN(エージーテック)-ファイルコンバータ(汎用 IBM↔MS-DOS) ……… Ⓜ¥42,000 DTCONV 付Ⓜ¥52,000
■ WORD STAR(マイクロプロ)-定評ある英文ワープロソフト Ver5.0 ……… Ⓜ¥125,000
■ WORD STAR-LT(マイクロプロ)- PC-98 LT 用……… Ⓜ¥49,000
■ TWINSTAR(マイクロプロ) ……… Ⓜ¥165,000
■ SUPERP ROJECT -時間・コスト・人材をマネージメントしさまざまなジョブで使用可能 ……… Ⓜ¥165,000

8 Bit Software (CP/M 80)

□ Plink Ⅱ (Phoenix) - L 80では不可能な 64K バイトをフルに使ったオブジェクト作成が可能。Plink86plus 同様オーバーレイはもちろん、アドレスの自由配置、シンボル情報の出力などが可能。但し、CPU は Z80に限る ……… ¥85,000
■ WORDMASTER(マイクロプロ) ……… ¥38,000
□ XASM シリーズ (AVOCET) 以下のターゲット用ラインアップ ……… 各¥67,500
● XASM-05、XASM-09、XASM-18、XASM-48、XASM-51、XASM-65、XASM-68、XASM-Z8、XASM-400、XASM - F8、XASM-04、XASM-85、XASM-Z80、XASM-180、XASM-75、XASM-11
□ XMAC-68K (AVOCET)……… ¥140,000
■ TKW-ZBC ……… ¥45,000
■ Hi-TECH C(ハイテック)……… ¥47,500
■ Rgy FORTH(リギー・コーポレーション)
● Fifth 180(Z80) ……… 各¥90,000
● XFifth 86、XF8000 ……… 各¥90,000
■ WORDSTAR(マイクロプロ) ……… ¥98,000

上記金額に消費税は含まれておりません。お買求めの際は上記金額に3%がかかります。

価格及び仕様は予告なく変わる場合がありますので、あらかじめご了承ください
マニュアルはソフトウェアと併せて販売して居りマニュアルのみの販売は致しません
又ここに載っていないソフトウェアも取扱っておりますので、お問合せ下さい

株式会社ソーテック
工人舎事業部
☎(045)662-0688(代)
本社:〒231横浜市中区太田町4-55 横浜馬車道ビル FAX.(045) 662-0656

Cross Debugger-MS v2.0

スクリーンモードで手軽なデバッグ、
教育やCPUの勉強にも威力を発揮します。

Cross Debugger-MSがバージョン2になりました。従来の機能はそのままで、スクリーンモードがさらに使用できます。シュミレーション時のレジスタやメモリの変化を常に監視できます。メモリは2つのブロックと、10個の変数(アドレス)を同時に監視できます。

現在状態の表示
現在設定している4つのブレークポイントの数と、ヒストリの状態をしてロードアドレス域の表示を行います。

メモリブロック1・2
連続したメモリブロックを表示します。スクリーンエディット形式で内容の変更も簡単です。2つのブロックを1つにして連続160バイトのメモリ表示もできます。もちろんシュミレーション実行中にメモリ内容の更新を行います。

メモリ監視
最大10箇所のアドレスを監視します。シンボル名・アドレス・内容を1行で表示します。普通は変数のカウンタなどの監視に使用します。

コマンド入力
コマンドを入力するエリアです。このエリアは9文字分のスペースしかありませんが、それ以上では横にスクロールします。

アセンブル表示
逆アセンブルの表示や、アセンブル実行用のエリアとして使用します。またシュミレーション実行時はこのエリアに表示します。

レジスタ表示
対象CPUの全レジスタとフラグのシンボル表示を行います。レジスタの保存は9つまでできますが、その状態の表示も行います。

ノーマルモード　スクリーンモードでもほとんどのデバッグ作業が可能ですが、コマンド形式のノーマルモードのメリットも見逃せません。例えばシュミレーション結果のプリンタへの印刷や、規定処理のバッチ実行などはノーマルモードの方が便利です。またノーマルモードはMS-DOSのV2.1以上が動作するマシンであれば実行できます。

名前	機能	摘要
SC	ヘルプメッセージの表示	NS
SC	スクリーンモードの切り替え	NS
LA	ロードアドレスの表示	N
D	メモリの16進ダンプ	NS
D2	メモリの16進ダンプ(エリア2)	S
M	メモリの表示と設定	NS
M2	メモリの表示と設定(エリア2)	S
FI	メモリへのデータセット	NS
C	メモリのコピー	NS
MC	メモリの比較	NS
F	メモリのサーチ	NS
L	ファイルとシンボルのロード	NS
LS	シンボルのロード	NS
SS	Sフォーマットでのセーブ	NS
SI	インテルHEXでのセーブ	NS
B	ブレークポイントの表示と設定	NS
BS	停止するブレークポイント	NS
BR	全レジスタを表示するブレーク	NS
BQ	指定レジスタを表示するブレーク	NS
BT	指定文字を表示するブレーク	NS
BC	ブレークポイントの解除	NS
K	シンボルの登録	NS
KA	アドレス順シンボルの表示	NS
KL	名前順シンボルの表示	NS
KC	シンボルの全解除	NS
V	メモリ監視エリアの設定	S
T	ステップ実行	NS
G	シュミレーション実行	NS
J	高速シュミレーション実行	NS
GS	全値を表示するシュミレーション	S
JS	PC値を表示するシュミレーション	S
R	レジスタの表示と設定	NS
RS	現在のレジスタ値の退避	NS
RL	退避したレジスタ値の復帰	NS
H	ヒストリのオン/オフ/表示	NS
HC	ヒストリの表示位置のリセット	NS
A	アセンブル	NS
U	逆アセンブル	NS
UF	ファイルに出力する逆アセンブル	NS
I	I/Oエリアのダンプ表示	NS Z80
O	I/Oエリアへの書き込み	NS Z80
#	計算	NS
X	ファイルでコマンド実行	NS
	MS-DOSコマンドの実行モード	NS
Q	終了	NS

N……ノーマルモード　S……スクリーンモード　Z80……Z80版のみ

MS86-S03(6800,6801,6802,6803,6808,6301,6303対応)…………¥38,000
MS86-SZ80(Z80対応)……………………………………………¥38,000

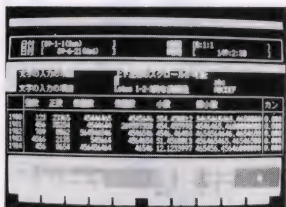
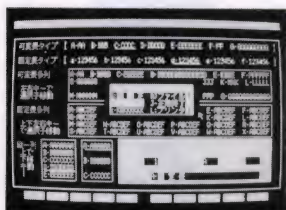
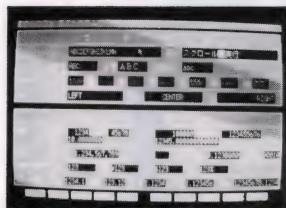
スクリーンモードはPC-9801, PC-286, FMRシリーズ, FM-16β, Panacom Mシリーズで動作いたします。ノーマルモードはMS-DOS V2.1以上が動作する機種に対応しております。登録カードをお送りいただいたユーザの方へはバージョンアップの通知をいたします。またお送りいただいてない方はバージョンアップができませんので至急お送りください。

資料請求No.100

強力な機能を追加して新登場

で〜ぐ3……………………………¥58,000

PC-9801(E以上)、PC-286で動作
MS-DOS V2.11以上およびバスマウスが必要です

キー入力・表構造・コマンド・アイコンなどのMMI処理をサポート

「で〜ぐ」で定義しCプログラムを生成

「で〜ぐ」で画面レイアウト、項目の定義を行い、Cのプログラムを生成します。そして提供するライブラリとリンクすれば左記の写真の様な入力を簡単に実行できます。

文字や数値の入力

画面作成だけでなく非常に幅広い形式の入力をサポートしました。文字はスクリーンエディット形式で、スクロールも行います。寄せ表示、漢字フロントプロセッサのサポートと各種の表現が可能です。数値は整数・正数・長整数・小数・倍小数を電卓形式かスクリーン形式で入力できます。

選択形式の入力

固定長・可変長そして縦横自由な選択形式の入力をサポートしています。必要なときだけ画面にポップアップして選択することもでき、スクロールも可能です。

カルク的な表形式の入力

伝票的な入力をサポートしています。縦や横にスクロールし、画面の有効利用ができます。

ツリー状のコマンドを実行

ユーザが定義したツリー状のコマンドを実行できます。コマンドは最終的にはユーザの関数か、コマンドのために用意した関数をコールします。選択だけでなく文字や数値のコマンドパラメータも自動的に入力できます。

マウスのサポートとアイコン動作

項目の移動にマウスが使用できます。また画面エリアを定義すればそこをクリックしたときの動作を指定できます。

CROSS ASSEMBLER-MS

マイクロとクロスリファレンスが使えます。

Cross Asembler-MSは、完全な2パス形式のクロスアセンブラです。8ビット系CPUのほとんどのCPUをモーラしています。豊富な凝似命令、条件付きアセンブル、マクロ、クロスリファレンスと高機能です。ラベルは6500個まで使用でき、3万行以上のアセンブルが可能です。
特徴❶ラベルは10文字まで認識します。❷オブジェクトの出力はインテルHEXフォーマットか、モトローラSフォーマットです。スイッチでどちらでも出力できます。❸豊富な種類の条件付きアセンブルが行えます。IF、IFC、IFEQ、IFF、IFN、IFGTなどなど。❹凝似命令は80系はDB、DC、68系はFCB、RMBと80系と68系を区別しています。よってよりインテル、モトローラ表記に近い形での記述ができます。❺数値のの表現が豊富です。PrefixもSuffixも完全にサポートしています。例えば16進数のAB12は、「$AB12」でも「OAB12H」でも使用できます。❻非常に多くの演算子をサポートします。単項演算子の+、−、∧(NOT)、{(Highバイト)、}(Lowバイト)、二項演算子の+、−、*、/(加減乗除)、>>、<<(シフト)、=、<、>、<=、>=、<>(関係演算)、&、I(論理演算)など❼各種8ビットCPU用のアセンブラが揃っています。❽6500個までのラベルが使用できます。ラベル6500個はほぼアセンブルリストの3万行程度に相当します。オブジェクトでは60Kバイト程度であり、8ビットCPUでは限界値です。❾特定のCPUの配慮も忘れてはいけません。例えば6809の「SETDP」命令も、68系の強制的にダイレクト命令をエックステンド命令に変換する「<、>」修飾も使用できます。❿拡張命令を追加したCPUもサポートします。例えば6801のクロスアセンブラは6301の特殊なインヘレント命令や、メモリとイミーディエイトの操作もサポートします。⓫TTL、STTLそしてOPT凝似命令などが使用できますので、きれいなリストを出力します。⓬パス1、パス2における条件付きアセンブルが可能です。コードを生成しなくてラベルの定義のみしたい場合に便利です。⓭MS-DOSが動作していれば殆どのマシンで使用できます。⓮非常に安価でお求め易い価格です。

型　番	提供アセンブラ名	価　格
MS86-XXXX	各々のアセンブラのうち1つ	各¥28,000
MS86-80	8048/8051/8085/Z80/Z8	¥58,000
MS86-68	6502/6801/6805/6809/6811	¥58,000
MS86-ALL	すべてのアセンブラのセット	¥80,000

CROSS C COMPILER

68032のCコンバイラが新登場

モトローラ系CPUのC言語としては、実績が非常に高いイントロール社の開発用Cコンパイラです。改訂を重ね今までに5回以上の大きなバージョンアップを行なっています。よって他のCに較べ、非常に洗練されたオブジェクトを生成します。そのオプチマイズの優秀性を高速で、最小サイズのオブジェクトコードが証明しています。ぜひCソースと、生成したアセンブルプログラムを比較して見てください。これはプロ仕様のマイクロコンピュータ開発言語です。
特徴❶カーニハン・リッチのCをフルにサポートし、ANSI規格のCコンパイラにコンパチブルです。初期化やビットフィールド、IEEEの浮動小数点などを完全にサポートします。さらにvoid, enum, volatile, constなどのANSI規格の型をサポートします。❷完全にROM化の可能なオブジェクトコードを生成します。データ部とコード部を各フェーズで分離しリンカがそのアドレスを決定します。❸最終的にモトローラS、インテルHex、テクトロニクスHex、拡張Hexの各フォーマットファイルを生成します。❹コンパイラ、リロケーティング・マクロアセンブラ、リンカ、ローダ、Hexフォーマッタ、ライブラリ・マネージャ、アブソリュート・リスティング・ジェネレータ、シンボル・テーブル・フォーマッタ、標準ライブラリ、実行時のマルチタスクのためのソースプログラムなどのパッケージを含みます。❺ユーザが作成したオブジェクトコードは、提供するライブラリを含んでいてもロイヤリティは必要ありません。ユーザが自由に使用、販売ができます。❻高度なオプチマイズにより高速で、最小サイズのオブジェクトコードを生成します。例えば6303CPUはスタック操作命令が少ない為コンパイラは苦労しますが、ローカル変数が少ないときは、PSHXでエリアを確保しますが、変数が多いときはSPをXに転送しXから必要なエリアを引きまたSPに返します。この様にCPUの特性をフルに利用し、また状況に応じた最適のコードを生成します。また自動的に最適なレジスタを使用し、必要なければメモリとのロード/ストアを避け、レジスタだけで処理します。❼価格はホストマシーンと、ユーザ数で異なります。MS-DOS用は48万円です。❽ソースレベルのデバッガが用意されています。またアセンブル以後が共通なモジュラIIもあります。詳しくはお問い合わせください。

名前	ターゲットCPU	種　別
C01	6801/6803/68701	8ビットCPU
C03	6301/6303	8ビットCPU
C09	6809/68HC09	8ビットCPU
C11	68HC11	8ビットCPU
C68	68000/08/10/12	16ビットCPU
C20	68020/30/68881/68851	32ビットCPU
C32	32016/32/32081/32082	32ビットCPU

本広告に掲載の商品の価格には消費税は含まれておりません。

手のひらサイズの
シングルボートコンピュータ
「グッピー」シリーズ好評発売中です。
カタログの請求はFAXでお願いします。

SMart SOFT MART, inc. ソフトマート株式会社
〒101 東京都千代田区神田須田町1-18-6 第1谷ビル ☎(03)256-5881 FAX(03)256-6180 GII GIII

TEAC

DATテープを採用した、
PC-9800シリーズ用の大容量サブシステム。
MS-DOS下でのオペレーションが可能で、
手にしたその日から、大容量を活用できます。

DATテープの採用で記憶容量、
1ギガバイト。

DATデータストレージシステム
RS-2 ¥398,000

■記憶容量、1ギガバイト

DAT技術の採用により、テープ1巻(TEAC DT-120Fデータテープ)に1ギガバイトの記憶容量を実現。大容量ですから、ファイルボリュームの大きい画像データや、各種計測データの収録用などに適しています。

■MS-DOSオペレーションが可能。

付属のデバイスドライバを使用することにより、MS-DOSコマンドで、直接RS-2を制御することができます。
また、フォーマットしたテープをセクタ管理しているため、データの追加記録や書き直しが可能です。

■1台のPC-9800シリーズに、2台のRS-2を接続可能。
■DAT標準フォーマットを採用。
■4ヘッドシステムによる高い信頼性。
■メディアの稼働履歴を管理。

その他のコンピュータ用周辺機器

ディスク&ストリーマ

40MB **DS-9840N** ¥428,000

20MB **DS-9820N** ¥328,000

カセットストリーマサブシステム
50MB **SB-2000** ¥218,000

22MB **SB-1000** ¥168,000

ファイルサーバー
80MB **FS-2** ¥998,000

150MB **FS-2** ¥1,398,000

無停電電源装置
AD-510 ¥120,000

プリンタコントローラ
1MB **PT-C 1MX** ¥89,000

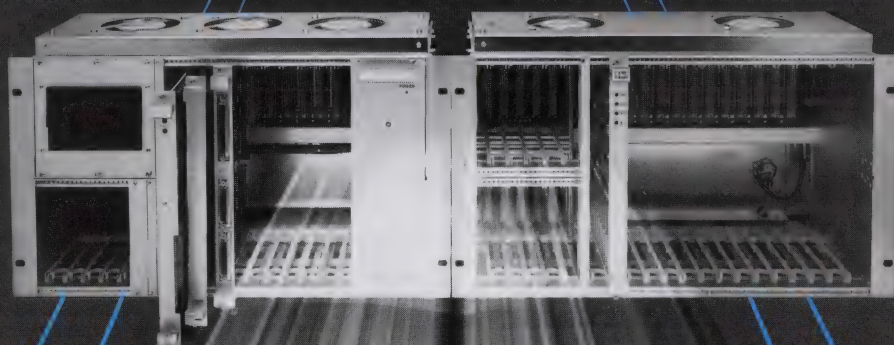
ACパワーディストリビュータ
AD-350 ¥35,000

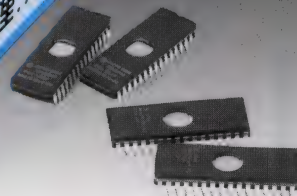
※価格はすべて標準価格です。(消費税は別)

ティアック株式会社
情報機器事業部・営業部 〒180・東京都武蔵野市中町1-19-18
コンピュータ営業課 ☎武蔵野(0422)52-5013
神奈川☎(0462)23-3903(代) 大阪☎(06)384-6041(代) 広島☎(082)294-4751(代) 仙台☎(022)227-1501(代)
茨城☎(0298)24-2865(代) 名古屋☎(052)782-4581(代) 福岡☎(092)441-3600(代) 札幌☎(011)521-4101(代)

資料請求No.102

シングルボード
コンピュータ
つきあい上手
V25CPUボード
V25－CPUボードは、CPUにNECのμPD70320L-8を採用したワンボード・マイクロコンピュータ。さまざまな構成が選択可能なため、幅広いアプリケーションシステムに利用できます。FDDは3.5インチまたは5インチが接続可能。また拡張インタフェースを使用し、各種インターフェースの接続が可能です。
※V25は日本電気株式会社の登録商標です。
●ブロック図
FDDドライブ 最大4ドライブ
LCD16×2 コントローラ内蔵型
4×6 TEN-KEY
電源ユニット 5V単一電源
FDC HD63265
LCDコントローラ インターフェース
TEY-KEY インターフェース
シリアル インターフェース
拡張 インターフェース
CPU μPD70320 (V25)
ROM 32K 最大128K
RAM 32K 最大128K
時計／カレンダ
不揮発メモリ 64バイト 最大256バイト
増設RAM 512K
(注意) 上記のブロック図は最大構成時の場合です。
RASSCO
ラスコ株式会社
本社/170東京都豊島区南大塚2-45-4　三栄ビル　TEL 03-946-8664(営業)

VME
各種ターゲットにお答えします。

OS-9/680XX
ソフト開発

FAターゲット組込み
NC ロボット 画像処理…etc.

●TVME®シリーズ一覧

TVME1002Aシリーズ	68000(10/12.5MHz)、68881(16MHz)、1MB/DRAM	TVME3200	インテリジェントGDC 68000+63484 RS-232C×1ch、キーボード
TVME1004Aシリーズ	MVME110アッパーコンパチ、カレンダー、バッテリー		
TVME1200Aシリーズ	68020(16MHz)、68881(16MHz)	TVME4000シリーズ	セントロ、RS-232C、ハーフサイズ
TVME2000シリーズ	DRAM、1MB/2MB	TVME6000	A/D 12bit、16ch、6.5μs
TVME2200	ROM/RAM28ピンソケット16ヶ、カレンダー	TVME9000シリーズ	バックプレーン:VME bus 9、14、20、10スロット I/O bus 5、9スロットI/O ch
TVME2400シリーズ	16/32bit SRAM 512KB/1MB/2MB(VSB対応)		
TVME3000A	セントロ、RS-232C×2ch	TVME9121	電源+5V30A、+12V7A、−12V1A
TVME3102	SCSI I/F、FDC(3.5、5、8インチドライブ可)	VXシリーズ	OS-9 16/32bit システム 19インチラック/デスクトップ/デスクサイドタイプ
TVME周辺	ラック、ケーブル他		

※OS-9はマイクロソフトの登録商標です。TVMEは橘テクトロンの登録商標です。

橘テクトロン株式会社 システム機器事業部
本 社 〒153 東京都目黒区東山2-2-5 日興パレス東山ビル ☎03(791)6631㈹ FAX. 03(791)1516
営業所 大阪06(304)0366㈹ FAX. 06(304)8110

資料請求No.106

SEKISUI

新製品

SCSIの到達点。

SCSI ANALYZER モニター/アナライザー

●SCSI MONITOR SC/1

SCSIラインの状態・データを解析する本格的ツール。オプションの活用でSCSI以外の各種標準インターフェースのタイミングモニターにも対応します。

●最大4MB/secの非同期・同期転送ラインをモニター。●フェーズモニター機能はSCSI規格コマンド全面対応。フェーズ翻訳やデータダンプでわかりやすく表示します。●豊富な設定が可能なフィルタ機能とトリガー機能で、観測は一層効率的。●最高10MHz/16ch同時観測のタイミングモニター機能は、波形表示観測、時間解析に威力を発揮します。●32Kステップのサンプリング・トレースが可能な大容量バックアップメモリー内蔵。●表示画面のハードコピー、測定データの連続記録が可能なプリントアウト機能。

●SC/1 メーカー希望小売価格215,000円

●SCSI ANALYZER SC/2

SC/2はSCSIシュミレーション機能を搭載。イニシェータ、ターゲット双方のシミュレーションが可能な1クラス上の高機能アナライザーです。

●モニター機能、プリントアウト機能など、SC/1の全ての機能をサポート。●SCSI機能レベル0〜2、SCSI規格コマンド全面対応。●送出データは、本機内蔵のプリセットパターンやユーザーが作成した任意のパターンが利用できます。●シミュレーション実行時データの エラーチェック・データ比較機能により、テスト対象機器のエラー発生頻度を観測可能。●テスト用フェーズパターンは、ステップ、連続シーケンスいずれでも実行できます。●シミュレーション結果はモニター機能で解析可能。

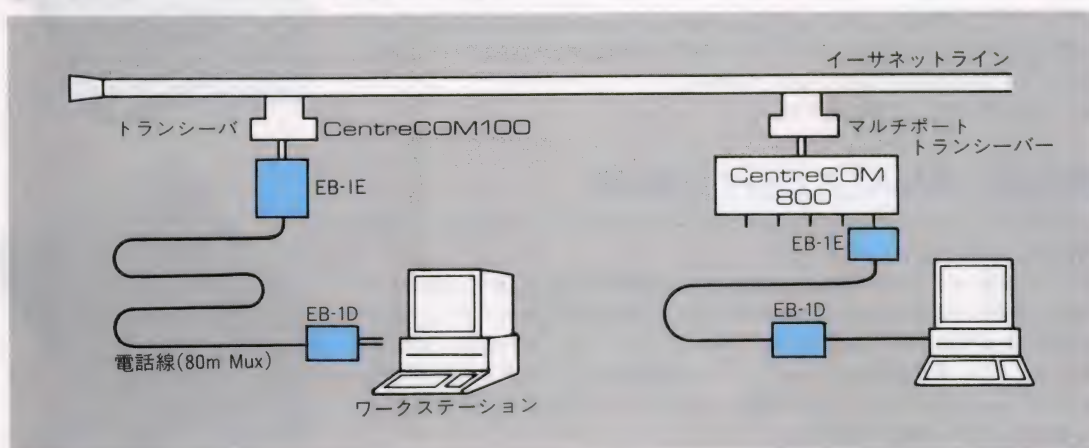
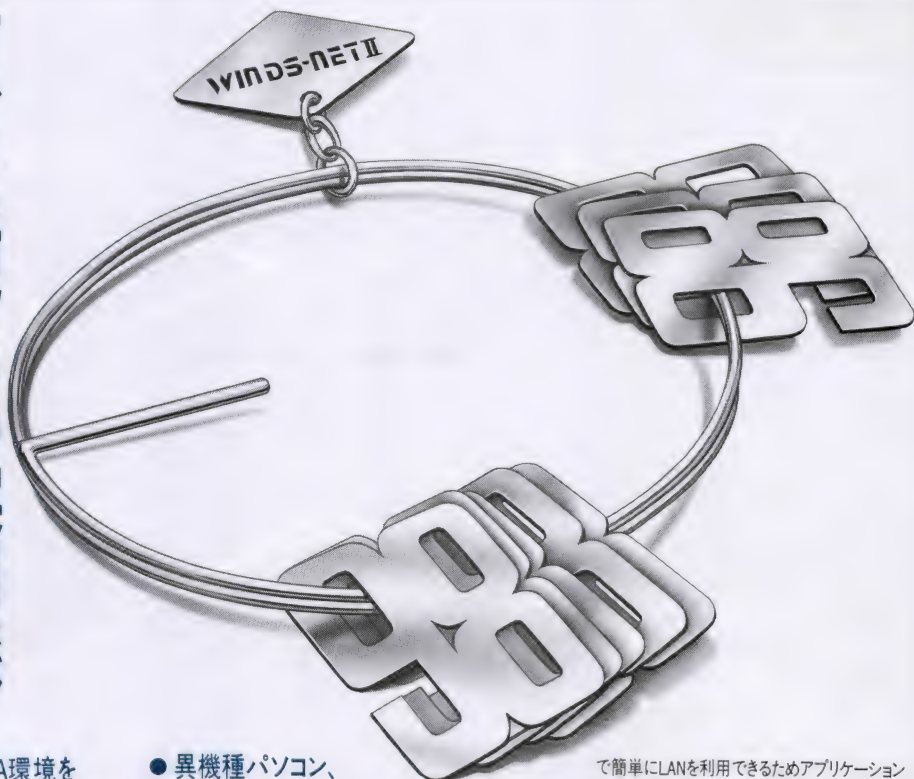
●SC/2 メーカー希望小売価格315,000円

●掲載商品に関してはご購入の際、別途消費税が付加されます。

積水化学工業株式会社
マイコン機器事業部

大　阪〒530 大阪市北区西天満2-4-4 TEL.06(365)4504
東　京〒105 東京都港区虎ノ門3-4-7(虎ノ門36森ビル) TEL.03(434)9109

資料請求No.108

NEC パーソナルコンピュータPC-9800シリーズ用

LAN すぐに使える実用派

● 光ファイバーで快適FA環境を

本品は伝送ケーブルに光ファイバーを採用したSDLCループLANです。信頼性の高いFA環境を実現するために、電気的なノイズの影響を受けず高品質のデータ送受信ができるように設計されています。

● インターフェイスボードがインテリジェンスに

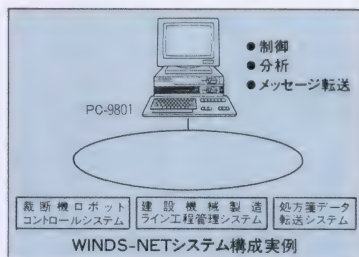
■ボードに搭載したCPU HD64B180が独立してネットワーク管理を行うためホスト(PC-9800シリーズ等)の負担を減らします。

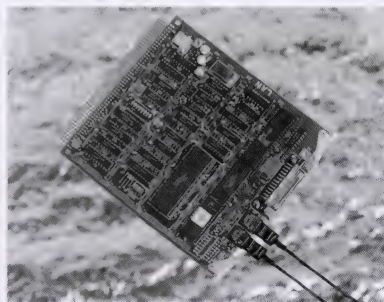
■255バイト×99フレームのバッファを持っているためホスト(PC-9800シリーズ等)が別プログラム実行中でもボードは他のノードからのデータを受け付けています。また、データを1～255バイトの可変長で送ることにより無駄のない能率的な通信が可能です。

■インテリジェンスだから機器構成もシンプルに。I/FボードをPC-9800シリーズの拡張スロットにセットして光ファイバーをつなげるだけでLANの完成です。

● 異機種パソコン、異装置ともフレンドリー仕様

異機種パソコン、異装置でもRS-232Cコネクタさえ持っていれば、オプションのRS-232C LANBOXでLANに接続してデータの送受信が行えます。

WINDS-NETシステム構成実例

● ソフト作りも楽々、時間と経費を節約します。

PC-9801～LANボード間のインターフェイスソフト標準装備。BASICやC言語プログラム中で、コールするだけで簡単にLANを利用できるためアプリケーション作成の時間が節約できます。

WINDS-NETⅡ
体験キット貸出中！

●キット内容：ボード2枚、光ファイバーケーブル

FA用光LANボード
WINDS-NETⅡ
NEW MODEL

■仕様: ●LAN構造/ループ・ネットワーク ●伝送制御方式/SDLCループモード ●データ伝送方式/周波数変調方式/FMO方式 ●伝送ケーブル/光ファイバー(HPCF) ●ノード数/最大16台 ●ノード間距離/最大1km ●伝送速度/192,000bps(可変長伝送方式)

¥148,000

▶御注文・お問合せ

推奨光ファイバーケーブル・コード
'TORAY'
ハードクラッド石英光ファイバ HCシリーズ
東レ株式会社 光ファイバ事業開発室
〒103 東京都中央区日本橋室町2-2-1
☎(03)245-5185・5186

―― サポートセンター ――

〒559 大阪市住之江区南港東8-2-66 日進ビル ☎(06)613-1313 FAX(06)613-5850

―― 販売・お問合せ ――
●大阪 八洲電業株式会社
〒537 大阪市東成区中本4丁目13-3
☎(06)972-3045 FAX(06)974-0633
●東京 株式会社日進電機製作所 東京営業所
〒108 東京都港区高輪4丁目19-11-401
☎(03)442-2094 FAX(03)442-2287

サポートセンター・各お問合わせ先へのファクシミリによる資料請求も歓迎いたします。

FA、通信システムはおまかせ下さい。
PC-COMとELXの絶妙な組合せが多くの問題を解決してきました。

リアルタイム／マルチタスクモニタ

FA、通信システムの中核OSとして、生産ライン監視、ロボット管理、ビル管理、各種プラント制御、計測システム、通信システム、データロガなど、幅広い用途に使用することができます。

ELX-286
- for PC-9800(286CPU/386CPUシリーズ)
- プロテクトモード

ELX-86M PC9801
- for PC-9800シリーズ
- 走行可能なハードウェア：PC-9800、FC-9800シリーズ パーソナルコンピュータ
- FA・通信システムに数百社の使用実績で実証
- 各種I/Oドライバが準備されています

FC-RTモニタ(86)
- for FC-9800シリーズ
- FC-RTモニタ(86)はファクトリコンピュータFC-9800シリーズ用リアルタイム・マルチタスクモニタで日本電気(株)の製品です。
- エルミックシステムでは、FC-RTモニタ(86)をC言語ライブラリ及びサポートとセットで販売しております
- 各種I/Oドライバが準備されています

ELX-86M FMR 新製品
- for FMRシリーズ、パナコムMシリーズ

ELX-86M/AT
- for IBM PC/AT、AX(EGA対応)

ELX-68K SBC
- for VMEボード
- 下記の環境で開発することができます：CP/Mマシン・パソコン/MS-DOS、CP/M-86・VAX/VMS、UNIX・YHP社HP-9000・立石電機社SX-9100

ELX-86M SBC
- for 御社シングルボード、マルチバスボード（御社のボードで是非お試しください）

インテリジェント通信制御ボード

パソコン本体に負荷をかけずに通信制御がおこなえます。
「公衆回線通信」、「ホストコンピュータ通信」、「パケット交換網(X.25)通信」、「各種生産、計測装置通信」、「シーケンサ通信」、「半導体製造装置通信」などの通信を、インテリジェント通信制御ボードは可能にします。

PC-COM V50
- X.25
- HDLC(ABM)、HDLC-BA(80)、LAP-B

PC-COM Z80
- TTY手順(無手順)
- V25bis手順
- ベーシック手順
- BSC1手順
- BSC3手順
- L2A(レベル2A)
- L2B(レベル2B)
- SECS手順
- シーケンサSysmac手順
- 三菱シーケンサ手順

FM-COM/Z80 新製品
FM-COM V50 新製品
AT-COM/Z80
AT-COM/V50

- IBM PC/ATはIBM社の、CP/M、CP/M-86はデジタルリサーチ社の、MS-DOSはマイクロソフト社の、VAX、VMSはDEC社の登録商標です。UNIXオペレーティングシステムはAT&T社が開発しライセンスしています。
- ELX-86M/PC9801は弊社がPC-9800用に販売している本格的なリアルタイム／マルチタスク・モニタです。
- 手順ソフトウェアの開発も行っておりますのでご相談下さい。

株式会社エルミックシステム
本社 〒231 横浜市中区弁天通4-59 第一生命ビル ☎045(664)5171(代)
大阪事業所 〒550 大阪市西区北堀江1-3-3 モーリグランドビル ☎06(541)3695(代)
盛岡事業所 〒020 盛岡市神明町5-5 岩手県火災共済ビル ☎0196(54)7531(代)

お問い合わせは下記まで。
営業技術部 ダイヤルイン ☎045-664-5171
大阪事業所 ダイヤルイン ☎ 06-541-3695
盛岡事業所 ダイヤルイン ☎0196-54-7531

資料請求No.110

資料請求No.112

8チャネルの切り換えをソフトで制御!!
低価格のRS-232Cセレクタ。

コンピュータ機器のネットワーク化を図る場合に障壁となるのが、まるでタコ足のような複雑な配線です。
「オクトパス」はRS-232Cをもつ各種の周辺機器を自由に接続できる高性能セレクタです。その上、手のひらサイズのコンパクト設計で魅力の低価格/
ケーブリングの煩わしさを一挙に解決し、接続の切り換えをスムーズに行うことができます。

■主な仕様
- ●チャネル数 RS-232C 7チャネル
  セントロニクス 1チャネル
  (標準I/F、PC98用I/F 2種類対応)
- ●切り換え対象 TxD、RxD、GND(3線式)
- ●標準付属品 ●専用2mケーブル3本(DTE2本、DCE1本) ●電源アダプタ
- ●ケーブル 6線ケーブル 最大延長距離10m
- ●電源 AC100V 消費電力5W
  (専用電源アダプタ使用)
- ●外形寸法 W100×L180×H30(mm)/620g
- ●ロングブレーク信号送信後のチャネル切り換え作業域〈コマンドモード〉での各種パラメータ
  ＊接続機器間の通信には関係ありません。
  - ●通信方式 調歩同期式(非同期)
  - ●通信速度 9600、4800、2400、1200、600、300(bps)
  - ●コマンドコード ASCIIコード
  - ●データ長/パリティ 7(bit)/あり(偶数・奇数)、8(bit)/なし
  - ●ストップビット 1(bit)または2(bit)

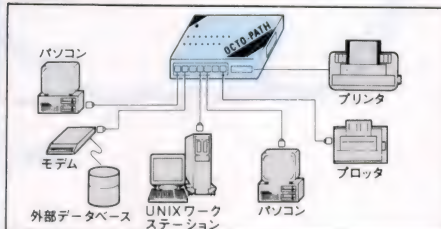
開発・販売元
株式会社 マックスブレイン
〒600 京都市下京区烏丸通り綾小路下ル四条地下鉄ビル5F ☎075(341)4801 FAX075(341)0674

取扱店
ソフトプロダクトセンター
東洋エンジニアリング株式会社
☎03(222)0541 FAX03(222)0545

MAXBRAIN

特長

- CPU内蔵により、切り換え操作はソフトで制御(コマンド方式)
- コマンド呼び出し方法は、接続機器からのロングブレーク信号受信による。
- RS-232Cを7チャネルとセントロニクスI/Fを1チャネル接続可能。
- チャネル毎にニックネームの設定可能。
- シリアル/パラレル変換、パラレル/シリアル変換可能。
- バッテリー保持により、電源断時も接続状態は保存。
- 専用ケーブルなので、オクトパス側コネクタはモジュラージャック、接続機器側コネクタはDSUB25ピン、ケーブルは6線ケーブルを利用。
- オプションケーブルを利用すると最大10mまで延長可能。
- 1:7(ブロードキャスト)、4:4(並列処理)、etc.用途に合わせて8チャネルを自由に組み合わせて利用可能。

●接続例1

■用途例
- 数台の端末を接続して簡易LANを構築する。
- 数台の端末でRS-232C仕様製品(プロッタ、モデム等)を共用する。
- 工場等でRS-232C仕様製品の品質管理試験を行う。
- RS-232C出力データをプリンタに印字する。
- プリンタ出力データをRS-232C上に取り込む。
- デージーチェイン設定により数多い機器を利用する。

コンパクトボディ・ハイパワーRS232Cセレクター
OCTO-PATH
〈オクトパス〉 ¥29,800 (標準セット 税別)

■オプション
- ケーブルの追加、距離の延長を行いたい場合に各種準備しています。
  ケーブル(DSUB25ピンコネクタ付)2m〜10mまで
  標準2m 2,000円(2m延長毎に1,000円加算)
  フリータイプ2m 1,500円(2m延長毎に1,000円加算)
- DSUB25ピンコネクタの端子の設定を独自に行いたい場合に使用します。
  コネクタ配線治具 2,000円
- ロングブレーク信号の送信ができない機器を使用している場合にハード的に送信することができます。
  接続機器(端末)とオクトパスの間に接続します。
  ブレークボックス 3,000円
- ロングブレーク信号をソフト的に送信することができます。
  接続機器(端末)で実行します。
  切換制御ソフト ●PC98/FM-16用(5"2HD) 3,000円
  ●J-3100用(3.5"2DD) 3,000円

8月発売
プロテクトモード・サポート
MKP286
for 80286、80386
(PC-9801対応)
メモリ不足の解消、ROM化、移植に最適
ユーザにプロテクトモード機能を全面解放
★63年度IPA(情報処理振興事業協会)委託開発商品
MS-DOSのプログラム
をそのまま実行できる
CZAR86
for 8086、V30、V33…
(PC-9801対応)
MS-DOSエミュレータ、マルチウィンドウ、豊富なコマンド群を用意し、簡単なリアルタイム構築に最適
※8月より値上げ予定
好評発売中
コンパクトなリアルタイムモニタ
MKP86
for 8086、V30、V33
(PC-9801、J3100、IBM-PC/AT対応)
ROM化、移植に最適
発売4年で1000台以上の実績
★59年度IPA(情報処理振興事業協会)委託開発商品
★61年度通産省「優秀情報システム」受賞
マルチタスクOS
リアルタイム
計測システム
メカ制御
工程管理
データ・ロギング
通信制御
〈ファイルコンバータ〉 価格58,000円
ヒューレットパッカード社 ⟷ PC-9800(MS-DOS)
(LIF)
HP64000など
(5インチ2D)
Vm2 Vm21/VX2/VX4、
(5インチ2HD)
◎MKPシリーズは、全て、MS-DOSとの親和性を大切にしているリアルタイムOSです。
初めてリアルタイム分野に挑戦される、プログラマ/SEをフレンドリなサポートで応援します。

ACCESS

UNIXでクロス開発を!!

CP/Mエミュレータ DUET-80

DUET-80は、CP/MアプリケーションをUNIX上でお使いいただくためのソフトウェアエミュレータです。UNIX上で、CP/Mアプリケーションを実行できるだけでなく、CP/Mの環境を実現することもできますので、簡単で、しかも優れたクロス開発環境を提供致します。
完全にソフトウェアのみでZ80CPUを含めたエミュレーションを行いますが、4〜5MHzのZ80相当の速さでCP/Mのアプリケーションを実行していただけます。SONY NEWS用に続きOMRON SX-9100シリーズ用も新たに加わりました。

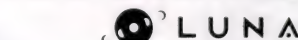
〔主要機能〕

1)CP/Mのアプリケーションを実行できる。
　従来からあるCP/M上の開発ユーティリティを無駄なく使える。
　CP/M用に作成したモジュールを実行(テスト)できる。
　……………………… UNIX上でi8080/Z80用のアプリケーションを開発可能
2)UNIXコマンドをDUET-80上から実行できる。
　DUET-80で実現したCP/M環境の中からUNIXの各種コマンドやユーティリティが実行できる。……………………… 開発効率の向上
3)CP/Mマシンとファイルの転送ができる。…… ファイル転送用ユーティリティが付属

《SONY NEWS用》 ¥280,000
《OMRON SX-9100シリーズ用》

XMACRO-80 クロス・マクロ・アセンブラ

XMACRO-80は、8080、8085、Z80用のオブジェクトモジュールを開発するためのクロスソフトウェアパッケージです。アセンブルでは、大容量ソースファイルが取り扱え、インテル社準拠のマクロ機能をサポートし、マイクロソフト社準拠のリロケータブルファイルを出力します。リンクは、シンボル数に制約がなく、64KByteのモジュールの作成が可能です。また、シンボリックデバッカ(XSID)も用意されており、効率の良いデバッグが可能です。SONY NEWS版に続きOMRON SX-9100シリーズ版が加わり、クロス開発環境をより強力にサポート致します。

《SONY NEWS版》
★XMACRO-80Package＋XSID-Z80 or XSID-85 ¥380,000
＋XSID-Z80 and XSID-85 ¥480,000
★XSID-Z80 or XSID-85 ¥180,000

《OMRON SX-9100シリーズ版》 LUNA
《MS-DOS版、CP/M-86版》 ¥120,000〜

CONCERTO(コンチェルト) OS統合化アダプタ

- 8MHzのZ80を使用しているため高速実行が可能。
- PC80S31用ポートを実装。汎用I/Oポートとしても使用可能。
- ボード内部に64KByteのRAMを持ち、98のメモリ空間は任意の2KByteのみ使用。
- 拡張用としてバスを外部に引き出すことも可能。

Zエンジン

CP/M-80アプリケーションを高速に実行させるため、Z80CPUボード「Zエンジン」をCONCERTOに搭載致しました。このZ80CPUボードは、CONCERTO上からサポートされており、組み合わせて御使用いただくと、よりスピーディな開発が望めます。

CONCERTOは、各種OSファイルを統合的に取り扱うことのできるソフトウェアです。
MS-DOS上でCP/M-80, CP/M-86, オプションでCP/M-68Kをエミュレートします。各OSエミュレータの起動は自動的に行うため、CP/MのアプリケーションをMS-DOSのアプリケーションと同様に実行することができます。また、MS-DOSのパイプやリダイレクションがCP/M-80, 86のアプリケーションにも使用できます。更に、CONCERTOでは、MS-DOS上でCP/M, N88BASIC等のファイルシステムをMS-DOSのファイルと同様に扱うことができるようになります。各OS間のファイルコンバートは、相方向に可能であり、その際ファイルを自動判別します。この他、CONCERTOは、UNIXのC-shellに準拠したヒストリ機能、エイリアス機能、ホームディレクトリ機能などもサポートしております。CONCERTOは、開発環境を広げる数々の機能をそなえております。

CONCERTO
PC-9801シリーズ版 **RA、RX対応**
(MS-DOS V2.11またはV3.1が必要)……¥68,000
PC-98XA版 **XL、RL対応**
(MS-DOS V2.11またはV3.1が必要)……¥88,000
FM-16β版(MS-DOS V3.1が必要)…¥88,000
★PC-98XA版、FM-16β版はPC-9801シリーズ版とは一部内容が異なっております。

CONCERTO PC-9801シリーズ版オプション
CP/M-68Kエミュレータ
CPEM68K ……………………¥38,000
●CONCERTO(PC-9801シリーズ版)、PC-9801-16(68000ボード)、68000ボード用増設RAM128K Byte以上が必要。CP/M-68Kは付属しておりません。

Zエンジン(Z80 CPUボード)
搭載CONCERTO
★CONCERTOオリジナル版とは一部内容が異なります。

CONCERTO-RS
(PC-9801シリーズ版CONCERTO＋Zエンジン)
……………………¥98,000

CONCERTO-RSX
(PC-98XA版CONCERTO＋Zエンジン)
……………………¥118,000

※この商品価格には消費税は含まれておりません。
※MS-DOSはマイクロソフト社、CP/Mはデジタルリサーチ社の商標です。
　UNIXは、米Bell研の開発したOSの名称です。
※製品の仕様、名称は予告なく変更する場合もございますのであらかじめご了承ください。

有限会社アクセス 〒101 東京都千代田区神田神保町1-64 神保町協和ビル7F ☎03(233)0200(代) FAX.03(291)7019

資料請求No.117

FX68000-05
I/Fボード **GPIB**

●ボードサイズ/VMEシングルハイト●電源/+5V単一●コントローラー/μPD7210(NEC社製)を使用●チャネル数/1chIEEE-IB用コネクタ使用

VMEバス規格に準拠したGPIB I/Fのボードで、IEEE Std.488-1978で規定されている全インターフェイス機能を備えています。また、FIFOメモリの搭載によりシステムのスループットの低下を最小限に押さえることができます。　¥79,000

合わせてどうぞ。

FX68000-06
I/Fボード **SCSI**

●ボードサイズ/VMEシングルハイト●電源/+5V単一●コントローラ/WD33C93(ウエスタンデジタル社製)●DMAC/μPD71071C(NEC社製)

VMEバス規格に準拠したSCSI I/Fボードで、ANSI SCSIX3T9.2に準拠した機能をもっています。512Kbyteのバッファは、SCSIバスとの転送をDMA転送することができます。　¥97,000

FX68000V

お客様のニーズに応えて誕生したFX68000V。OS-9/68000(Ver.2.2)を搭載。シングルハイトボードを5枚まで収納。一段とパワフルになったハイレベルバージョンです。
¥698,000

ラスコ株式会社

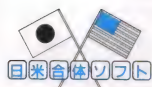
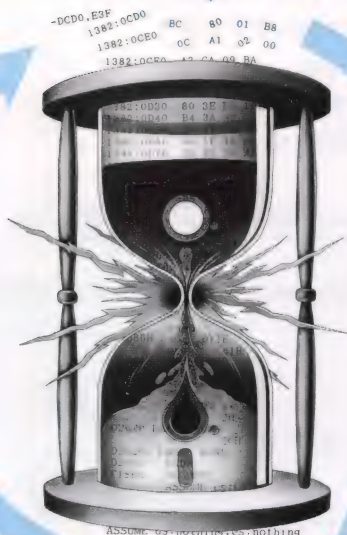
本社/170東京都豊島区南大塚2-45-4　三栄ビル　TEL 03-946-8664(営業)

※OS-9はマイクロウエアー社の登録商標です。

●取扱店リケイ電子株式会社本社/〒113 東京都文京区湯島1-1-2(ATMビル)　TEL03(257)1355代 FAX03(255)7036　担当 杉崎・須佐
●取扱店ハード&ソフトNDK　〒550 大阪市西区西本町1-5-3(扶桑ビル1F)　TEL06(543)0808 FAX06(543)2266　担当 浅野・蔡

前作『IQ輪廻』を大幅にグレードアップ!

80286の全命令に対応!

凄みを増した

▶5インチ2HD▶3.5インチ2HD▶8インチ2D▶5インチ2DD版

各¥32,000(送料共)

●全MS-DOSマシン用
(本体メモリ384KB以上、漢字ROM必要)
MS-DOSVer2.XX/3.XXで
ご使用ください。

用途
★実行ファイルの開発支援ツールとして
★マクロ/逆アセンブラの教育研究用として

『輪廻286』は、MS-DOS上で動作する80286用ソースジェネレータです。

輪廻286で実行ファイル(最大250KB位まで)をリターン・ポン。自動的にラベルが付き、ソースファイルを出力します。

このソースファイルは、コード/データの各エリアに区分されており、マクロアセンブラへ直接入力できます。

輪廻286の自動出力したファイルを、再度アセンブル。実行ファイルへ戻しても元と同じように動作するほど強力です。

★輪廻286特長★

❶**80286命令対応** CPU80286/80186の全命令に対応!
❷**強力自動出力** 自動モードでも、高い確率で正確なソースファイルを出力!
❸**大ファイルOK** 約250KB位までのファイルを1パス(リターン・ポン)で、ソースファイルを自動作成。
❹**マクロアセンブラ準拠** 出力ファイルをマクロアセンブラへそのまま入力OK!
論理セグメント、データ型式、オフセットアドレスをラベル付きで自動出力。
❺**エントリーポイント指定可** 入力ファイルの開始アドレスを複数箇所指定可能。
❻**リダイレクトOK** エントリーポイント入力や出力メッセージ等をリダイレクトできます。
❼**高速処理** 約20KBの入力ファイルを、30秒前後で自動出力。(RAMディスク使用時)
その他、スイッチ指定により、出力ファイルのorg、jmp等の命令を1TABインデントできます。

リダイレクトによるエントリーポイント指定可能!

※MS-DOS、MACRO ASSEMBLERは、米国マイクロソフト社の商標または製品名です。※MS-DOSシステムは内蔵していません。※製品のデザイン、仕様、価格等は予告なく変更することがあります。

MICRO DATA 株式会社マイクロデータ
住所 〒169 東京都新宿区高田馬場1丁目17番8号
☎03-RS-232C・PC-9801(代)

ESP-5000シリーズ新発売!

第1弾 ESP-5110 4M-DRAM

FMR/PANACOM　のメインメモリを拡張するモジュールです。
FMR　I/Oスロットに装着
PANACOM　I/Oスロット又はメモリ専用スロットに装着
FD/HDタイプ共用、パリティチェック付

ESP-2000シリーズ好評発売中!

(NEW) ESP-2220 OPT16 INT

- 入力点数　16点（各入力毎にフォトアイソレート）
- 入力形式　電流駆動入力
  フォトカプラ　PS2041
  フォトカプラ入力電流 外部より供給
  印加電圧　+5V～48V
  印加電流　5mA/点
- 入力処理（ジャンパによる選択）
  (a)フォトカプラの次段にバウンシング撤去回路、カウンタタイマによるハードウェア処理
  遅延時間 約30msec（出荷時）
  (b)バウンシング回路のバイパス
- I/O コネクタ　40ピン 半田付けタイプ

(NEW) ESP-240X SIO

- 通信LSI　Z80 SIO 2ch
- 通信形式　同期、調歩同期(非同期)
- 通信仕様　RS-232C 2ch ESP-2400
  RS-422(RS-485) 2ch ESP-2401
  RS-232C/RS422 (RS485) 各1ch ESP-2402
- 通信速度　非同期(50～19200bps)
  同　期(800～307200bps)
  各チャンネル毎に設定可能
- I/Oコネクタ　26ピン フラットケーブルタイプ×2

(NEW) ESP-2820 LED DRV.

- ドライバ　ICM7218×2
- 接続可能LED　7segLED（アノードコモンタイプ）
- ドライブ能力　8桁×2
- I/O コネクタ　16ピンフラットケーブルタイプ×2
- ベゼル付の専用LED ディスプレイ ESP-2906-4/6/8があります。

ESP-2000シリーズのファミリー

型番	名称	内容
ESP-2010	CPU-I	TMPZ84C 015AF-6 SIO.PIO.CTC.CLOCK.WATCH DOG TIMER
ESP-2200	OPT16 I/O	入力16点、出力16点、LEDによるモニタ
ESP-2201	OPT32 IN	入力32点、TLP521使用、バウンシング除去回路、LEDによるモニタ
ESP-2202	OPT32 OUT	出力32点、TLP572使用、LEDによるモニタ
ESP-2220	OPT16 INT	入力16点、Z80 CTCによる割り込み、PS2041使用、LEDによるモニタ
ESP-2300	PIO	Z80 PIO×2、4bit毎に入出力方向変更可、コントロールライン、I/Oバッファの変更可
ESP-240X	SIO	RS-232Cx2(ESP-2400)、RS422x2(ESP-2401)、RS232C/RS422(ESP-2402)
ESP-2600	AIO	A/D 12ビット(10μs)、16ch(差動8ch)、D/A:12ビット、10μs
ESP-2800	KEY/LCD	HD44780 専用、5×5 キーマトリックスに対応、ROM実装可、キー入力割込可
ESP-2810	PMC	2軸独立型、パルスモータ・コントローラ、PCL-240Kx2使用、1軸型(ESP-2811)
ESP-2820	LED DRV	7seg LED DRV 8桁x2、(ベゼル付専用LED DISPLAYあり)

型番	内容
ESP-2900	ICE 変換用ボード
ESP-2901	アナログ用ユニバーサル・ボード
ESP-2902	デジタル用ユニバーサル・ボード
ESP-2903	5×5 KEYマトリックス・ボード(ESP-2800用)
ESP-2904	セントロニクスI/F ボード(ESP-2010用)
ESP-2905-05/10/15	バック・プレーン5、10、15スロット
ESP-2906-4/6/8	ベゼル付LED ディスプレイ4、6、8桁(ESP-2820用)
CBL-202/203/204/205	接続ケーブル2連～5連
RACK-105/110/115	ラック(バックプレーン付) 5、10、15スロット
FPH-121	汎用フロントパネル
FPB-110/120	めくら板10、20㎜
ESP-9000	ESP-2010用シンボリックデバッガROM
ESP-9010	ESP-240X用シンボリックデバッガROM
ESP-9100	Remote Symbolic Debugger

ぷりんちゃん (V2.0)アプリケーションソフト対応版

FM R-50/60・Panacom M500 に異機種プリンタを接続!!

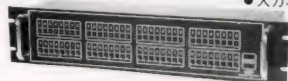
機種毎に異なるプリンタの漢字コード、罫線コード等を変換して出力する組み込み型のデバイス・ドライバです。文字品位、文字ピッチ、自動改ページ機能の選択ができます。登録、運用は全てメニュー画面よりおこなえます。

FM-R版　PC-9801版　J-3100版　16β版

FM-PR353	F9454xx	MB27410	PC-PR201	ESC/P24
FM-PANACOM	F9450/OPERATE	16β	PC9801	EPSON

対応アプリケーションソフトウェア　一太郎(ジャストシステム社)、MULTIPLAN (マイクロソフト社)、1-2-3(LOTUS社)
標準価格　¥30,000 (各版共通)：ぷりんちゃん PLUS　¥7,000

I/O MONITOR　好評発売中!!

ソフトウェアのデバッグに必要な入出力機器の模擬装置。
3機種 ● 入力64点(スイッチ)タイプSS ● 出力64点(LED) タイプLL ● 入力32点、出力32点 タイプSL

■仕様
スイッチ：接点容量 0.4VAmax(100mA、28Vmax)
L E D：シンク電流5mA[TYP.]以上必要
電　源：DC24V1.1A(約15mA/1点)内蔵
外部電源 DC5V～DC28Vを裏面端子より供給可能
サイズ：W480 × H99 × D125 標準ラック搭載可能

標準価格 ¥68,000-

※消費税は含みません。

汎用コンピュータ用インターフェース・エミュレータモジュール

- IBM及びFACOMの汎用コンピュータのブロックマルチプレクサ及びバイトマルチプレクサ・チャンネルに接続。
- 外部機器用としてセントロニクス・インターフェイスを実装。
- 大型機用エミュレータの設計・製作を承ります。

株式会社エスパー・システム
〒543 大阪市天王寺区生玉町19番16号(サメジマビル)
☎06-779-1315代　FAX 06-771-6657

取扱店　東亜無線電機(株) FAシステム部　☎06-644-6161
三協電子部品(株)東店　☎06-643-5567
シナダイン株式会社　☎東京 03-461-9311　☎大阪 06-304-7260　☎宇都宮 0286-59-3325

SACOM

無限の空間へ

高速・小型のハードディスクドライブ<Mocking Bird>新シリーズ

MB-45HT/45HR, MB-90HT/90HR

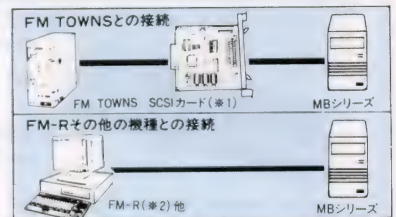
富士通・FM-TOWNS、FM-R/ナショナル・パナファコムMシリーズ/その他SCSIインターフェース採用機種に対応。

大容量

HDD3.5インチディスク一枚としては、現在のところ、世界最高レベルの45MB(フォーマット時)を実現。バリエーションとして45・90・135・180MBの四タイプを揃えました。

小型・軽量

超薄型3.5インチHDDを使用しており、外観はコンパクトにまとまり、重量も2.4kgと軽くなりました。パソコンの横に置いても気になりません。机の上も、できればすっきりとさせたいものですね。

		グレー アイボリー	MB-45HT MB-45HR	MB-90HT MB-90HR	MB-135HT MB-135HR	MB-180HT MB-180HR
記憶容量			45MB	90MB	135MB	180MB
平均アクセスタイム			25msec			
環境条件	動作時	温度	5°C~35°C			
		相対湿度	20%~80%(結露のないこと)			
	保管時	温度	−20°C~50°C			
		相対湿度	10%~90%(結露のないこと)			
入力電圧			AC100V ±10% 50Hz/60Hz			
消費電力			約19W			
寸法・重量			85(幅)×155(高)×240(奥行)mm 2.4~2.9kg			
付属品			接続ケーブル(SCSIケーブル)			

※1 富士通純正品FM30-121 ¥30,000(税別)
※2 FMR-30/30BXはI/O拡張ユニットとSCSIカードが必要

MB-45HT/HR(45Mバイト)……………¥180,000
MB-90HT/HR(90Mバイト)……………¥248,000
MB-135HT/HR(135Mバイト)………¥480,000
MB-180HT/HR(180Mバイト)………¥600,000

表示の価格は消費税を含みません。

株式会社 システム サコム
〒130 東京都墨田区両国4-38-16 両国桜井ビル TEL.03(635)5145 FAX.03(635)5148

●PC-9800シリーズ用●

GP-IBライブラリ

特長 1 豊富なGP-IB関数が用意されています。

創業以来、GP-IBによる自動計測プログラムを専門に開発してきた当社が、様々な計測機器を使用してきた経験から作り出した実戦的な関数群です。

C言語用：50個　QuickBASIC用：51個
N88BASIC用：20個　TurboPASCAL用：44個

特長 2 計測用タイマー関数が用意されています。

データのロギングや、計測のタイミングをとるために必要な10ms単位のタイマー関数が用意されています。

C言語用／QuickBASIC用 ……………………………10個

特長 3 実際の計測器を使用した応用例をソースファイルで提供します。

- C言語用の応用例
  YHPユニバーサル・カウンタ(HP5334B)、YHPインピーダンス・アナライザ(HP4192A)、アドバンテストデジタルマルチメータ(TR6846)、横河電機アナライジングレコーダ(3655E)、菊水電子デジタルストレージ・オシロ(COM7061A)、PC-9800間での通信
- QuickBASIC用の応用例
  アドバンテストFFT(R9211E)、アドバンテストデジタルマルチメータ(TR6846)、菊水電子デジタルストレージ・オシロ(COM7061A)
- N88BASIC用の応用例
  ソニーテクトロニクスデジタルオシロ(マイティー468)、小野測器トラッキングアナライザ(CF880)、WAVETEK任意関数発生器(model-75)、菊水電子デジタルストレージ・オシロ(COM7061A)、その他高速A/D変換器等
- Turbo PASCAL用の応用例
  アドバンテストデジタルマルチメータ(TR6846)、PC-9801間での通信、その他応用例8例

特長 4 Quick Cの統合環境に対応しました。

MS-C用ライブラリ(Sモデル専用バージョンは除く)では、Quick C用の"GPIB.QLB"を提供しておりますから、Quick Cの快適な開発環境でご使用いただけます。

特長 5 Cインタプリタへの組み込みが簡単にできます。

MS-C用には、C-terpへの組み込みのための関数一覧表ファイル".AUG" ファイルを提供します。Lattice C用には、Advanced RUN/Cへの組み込みのためにラージモデルのオブジェクト形式のファイルと".RCL" ファイルを提供します。

特長 6 引数チェックのためのヘッダファイルが付いています。

引数チェックのためのヘッダファイルが付属していますから、プログラミング時の引数記述ミスを防止できます。(C言語用、QuickBASIC用)

特長 7 C言語用には、実用性を評価していただくための安価なSモデル専用バージョンを用意しています。

対応パソコン：NEC　PC-9800シリーズ、(XA、XLハイレゾルーション・モードは除く)、EPSON　PC-286シリーズ
使用できるGP-IBボード：PC-9801-29/PC-9801-29K/PC-9801-29N
OS：MS-DOS　Ver2.1以上
メディア：8インチ2D、5インチ2HD、3.5インチ2HD
価格：
- C言語用
  MS-C用GP-IBライブラリ(Quick C用を含みます)…75,000円
  (Ver3.0～5.1 S、C、M、L、Hモデル対応。但し、OS/2では使用不可)
  Lattice C用GP-IBライブラリ ……………75,000円
  (Ver3.0～4.0 S、D、P、L、Hモデル対応)
  TURBO C用GP-IBライブラリ……………75,000円
  (Ver1.5～2.0 S、C、M、L、Hモデル対応)
  上記各商品のSモデル専用バージョン……………28,000円
- QuickBASIC用GP-IBライブラリ……………75,000円
- N88BASIC用
  MS-DOS版N88BASIC用GP-IBライブラリ…55,000円
  DISK N88BASIC用GP-IBライブラリ ………55,000円
- Turbo PASCAL(Ver4.0～5.0)用GP-IBライブラリ……75,000円

当社が、長年に渡り築き上げた自動計測のノウハウを、応用例のソースファイルとしてみなさまに提供します。提供させていただく12機種19例は、全て実機にて稼動確認を終了しております。

アドバンテストFFT(R9211E)	1例	エーアンドディFFT(AD3525)	3例
小野測器FFT(CF350)	3例	アドバンテストデジタルマルチメータ(R6551)	2例
YHPデジタルマルチメーター(HP3478A)	1例	フルークデジタルマルチメータ(8840A)	1例
アドバンテストマルチチャンネル・デジタルレコーダ(TR2724)	2例	インテックス2020型GPIBリレー	1例
日黒電波オーディオ・アナライザ(MAK-6600)	1例	菊水電子デジタル・オシロ(COM7061A)	1例
ヒオキMEMORY Hi CORDER(H8850)	2例	アドバンテスト直流電圧／電流発生器	1例

●メディア：5インチ2HD、3.5インチ2HD
●価格……………………………48,000円

■詳しくはカタログをご請求下さい。

価格に消費税は含まれておりません。
MS-DOS,MS-C,QuickBASICは米国マイクロソフト社の商標です。
Lattice Cは米国Lattice社の商標です。
Turbo C, Turbo PASCALは、米国ボーランド社の商標です
C-terpはGimpel Software及び㈱サザンパシフィックで開発されたソフト名です。
advanced RUN/Cは㈱LIFEBOATと米国Age of Reason社の共同開発製品です。

●パソコンによる自動計測システムの開発

〒470-01 愛知県愛知郡日進町赤池前田35 コヤマビル2F
TEL.(052)805-5177 ・ FAX.(052)805-5144

MS-C用GP-IBライブラリ代理店：
㈱シンフォニー
TEL. 03(443)0291

資料請求No.125

CT-9800B

PC-9801用
IMAGE PROCESSING BOARD

これがそもそものパソコン用フレームメモリー

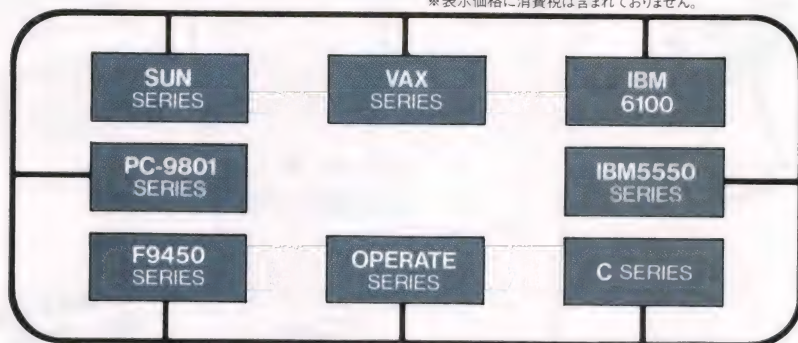
¥98,000

新たな挑戦、NET ACE誕生。

LAN

あなたのパソコンをLANに接続できる。

- イーサネットに接続可能
- TCP/IP telnet FTPをサポート
- 専用基板でハイパフォーマンスを実現
- 日本語サポート(DEC・EUC・S-JIS・JIS)
- VT100ターミナルエミュレータ「VTerm」を搭載

〈サポート機種〉(ハード・ソフト込み)
PC-9801シリーズ(XA, XLを除く) ¥248,000
IBM 5530/40/50/60 ¥248,000
F9450シリーズ ¥150,000
OPERATEシリーズ ¥150,000
Cシリーズ ¥150,000
※表示価格に消費税は含まれておりません。

CSK
●西システム開発センター: 〒540 大阪市中央区城見2-1-61ツイン21 MIDタワー35F Tel.(06)949-1050(代)
●東京・名古屋・九州・日立・広島
●お問合せは、株式会社CSK 西システム開発センター: 端山(ハヤマ)06-949-1050(大阪) OAシステム事業部: 古沼03-342-5299(東京)

資料請求No.127

あなたの 登録番号は？

小誌では、初めて資料請求された皆様全員に登録完了はがきで登録番号のご案内をしています。ところが、このはがきを紛失あるいは保管場所を忘れてしまった方もいるようです。

このときに、既に広告主より資料の送付を受けている場合ならば、その封筒の宛名ラベルから登録番号を見つけられます。ご氏名の下にある左側の10桁の番号が〈例：I012000345〉皆様個有の番号です。登録番号を使えば、簡単にそしてスピーディに資料がお手許に届きます。

広告主からの資料送付封筒など

資料請求会員登録完了の通知ハガキ（表）

資料請求会員登録完了の通知ハガキ（裏）

お問い合せは下記まで
CQ出版 広告部 インターフェース・広告
TEL 03(947)6311㈹・03(947)4949㈱

転機があったらすぐ転記

すでに資料請求会員登録を済ませた方が、昇進・転属・転職・就職・進学などをされて登録内容に変更が発生した場合は、下記にご紹介する方法でご連絡下さい。また、事務所が移転した時にも同様にご連絡下さい。
小誌のデータを修正します。
これで、会員の方は従来通りの会員番号で、従来通りの簡便さで資料請求システムをご利用いただけます。
ご協力をお願い致します。

資料請求カード　IF　1987年12月号

当社記入欄　1 VOL IF 1 2 7　2 受付
カード有効期限：1987年12月末日迄
送付先は勤務先に限ります(除学生)
3 住所コード
4 請求状況 1　1＝新規　2＝登録済　3＝登録内容の変更
5 会員番号 1012000345
6 姓名　スガモ 巣鴨　タロウ 太郎
7 生年月日 1951年 9月 9日
8 勤務先住所 170 東京都 豊島区 巣鴨 1-14-5 蘇ビル
9 会社名　部署名
10 勤務先電話番号 03 (947) 6633
11 役職　12 所属部門　13 担当分野
14 業種　15 従業員数

資料請求カード記入法

専門誌の広告は重要な技術情報です。その中で読者の皆様が入手したい情報は、資料請求カードのご利用で、居ながらにしてお手元に届きます。ぜひ有効にご活用下さい。
○必ず全項目にご記入下さい。
○姓名・勤務先・電話番号・住所は所定の様式通り1枠ごとに書き入れて下さい。
○会社名は株式会社ならば(株)、有限会社は(有)、財団法人は(財)と略し社名を書いて下さい。
○生年月日は西暦で書いて下さい。
○当カードは登録することができます。まず全項目にご記入の上投函下さい。当社より登録完了はがきが到着します。
○当社請求カードの会員番号がある方は、会員番号・姓名・生年月日の記入ですみます。また会員番号をお持ちで登録内容に変更がある方は変更箇所を上記の他に書いて下さい。
○役職・職種・専門分野・業種・従業員数は下記の該当する番号を記入して下さい。

- 登録状況欄に3とご記入下さい。
- 会員番号・ご氏名・生年月日を忘れずにご記入下さい。
- 変更箇所の番号に赤あるいは目立つ色で丸印を付けて下さい。
- 広告資料請求をされずに変更だけをご連絡いただく方は、広告資料請求欄に1Aとご記入下さい。
- 後日、新規登録の際にお届けした「登録完了はがき」を小誌からもう一度お送りします。

CQ出版
インターフェース・広告
TEL 03･947･6311(代)
03･947･4949(直)

資料請求カード記入法

専門誌の広告は重要な技術情報です。その中で読者の皆様が入手したい情報は、資料請求カードのご利用で、居ながらにしてお手元に届きます。ぜひ有効にご活用下さい。

- ○必ず全項目にご記入下さい。
- ○姓名・勤務先・電話番号・住所は所定の様式通り１枠ごとに書き入れて下さい。
- ○会社名は株式会社ならば(株)、有限会社は(有)、財団法人は(財)と略し社名を書いて下さい。
- ○生年月日は西暦で書いて下さい。
- ○当カードは登録することができます。まず全項目にご記入の上投函下さい。当社より登録完了はがきが到着します。
- ○当社請求カードの会員番号がある方は、会員番号・姓名・生年月日の記入ですみます。また会員番号をお持ちで登録内容に変更がある方は変更箇所を上記の他に書いて下さい。
- ○役職・職種・専門分野・業種・従業員数は下記の該当する番号を記入して下さい。

役職

1. 一般職
2. 主任職
3. 係長職
4. 課長職
5. 部長・次長職
6. 役員職
7. 経営者
8. 教職(小、中、高校)
9. 教職(大学、専門学校)
10. 専門職(弁護士・医師・会計士等)
99. その他

所属部門

1. 研究・開発部門
2. 設計・技術部門
3. 生産・生産管理部門
4. 保守サービス部門
5. その他の技術部門
6. 営業部門
7. 購買・資材部門
8. 総務・経理部門
9. 人事・教育部門
10. 経営部門
99. その他

担当分野

1. コンピュータ(ハードウェア)技術
2. コンピュータ(ソフトウェア)技術
3. 半導体・デバイス技術
4. 電子回路技術
5. 計測・制御技術
6. 通信関連技術
7. 物性・材料
8. 機構設計
9. システムエンジニアリング
10. 検査・品質管理
11. 営業
12. 経営管理
13. 各種サービス
14. 教職
15. 学生
99. その他

業種

1. 建設・不動産
2. 金融・保険
3. 化学・エネルギー
4. 鉄鋼・金属
5. 機械・精密機器
6. 電子機器・部品
7. 輸送用機器・運輸
8. その他の製造
9. 商社・卸売
10. 通信・情報
11. 情報処理
12. その他のサービス
13. 政府公共機関
14. 学校・研究所・病院
15. 農林・水産業
99. その他

従業員数

1. 10人以下
2. 11人～50人
3. 51人～100人
4. 101人～500人
5. 501人～1000人
6. 1001人～5000人
7. 5001人～10,000人
8. 10,001人以上

続OS-9/68K徹底解剖(上級編)

特集

▷先月号につづいて，OS-9/68K の特集である．OS-9/68K の特徴の一つは，個人ベースで移植でき，機能を拡張できる本格的 OS であるということである．そこで今月号では，とくにカスタマイズ & グレードアップをテーマとして特集を構成する．
▷移植は，いうまでもなく，デバイス・ドライバを作成することがそのおもな作業内容だが，論理的なファイル管理を行うファイル・マネージャとのインターフェース仕様にしたがって作成する必要がある．今回は，X68000 への移植事例を紹介する．
▷一方，機能拡張は，OS-9/68K の場合，ファイル・マネージャの新たな設計やカーネル自体の拡張機能の活用，などが考えられる．今回は，前者については，ファイル・マネージャの簡単な設計例を紹介する．後者については，リアルタイム性を向上させるために作成された拡張モジュール iREX を紹介する．その他，Internet への対応についても触れた．(編集部)

比較研究・OS-9とMS-DOSのファイル・システム

星　光行

OS-9/68K のファイル・システムについては，先月号の「OS-9/68K 再入門①」でも解説したが，ここではMS-DOSの場合と比較しながら，再度ディスク構造，ファイル構造について復習しておく．（編集部）

ディスク上のデータは，どんなOSでもデータのサイズが1セクタを超えると，当然ながら複数のセクタに分割して書き込まれます．この分割されたデータを何らかの方法で連続したデータとして扱うようにするのがDOSの大きな役割の一つです．これらのファイルの管理方法は，DOSによってそれぞれ異なります．そこで，参考までにパソコンでもっとも一般的なMS-DOSとマルチユーザ，マルチタスクのファイル管理を実現しているOS-9のファイル構造について簡単に説明します．

MS-DOSもOS-9も，ディスクに関する基本的な考え方は共通です．ただ，管理する方法が異なるだけです．具体的な説明の前に，この共通点をお話しておきます．

●論理セクタ

まず，両者ともディスクのアクセスはすべて論理セクタ番号で行われます．通常，トラック0，セクタ1を論理セクタ番号の0として，以下単純に論理セクタが連続して並んでいるものとして管理します．実際の物理的なトラック番号やセクタ番号はドライバ内で変換されます．こうすることによって，OSはディスクの種類やハードウェアの違いに関係なくファイル管理ができるようになります．論理セクタとは，OSがドライバに対してリード/ライトの要求を行う最小の単位のことです．MS-DOSでは，この論理セクタのサイズが512/1024/2048バイトの可変になっていて，必ずしも物理セクタ・サイズと同じとはかぎりません．

一方，OS-9では今のところ256バイト固定で，しかも，論理セクタと物理セクタのサイズが同じになっています．

論理セクタ長が1024バイトの場合，OSはつねに1024バイトの論理セクタ長でドライバを呼び出します．ディスクの物理セクタ長が1024バイトであれば，論理セクタ長と物理セクタ長が1対1で対応することになります．しかし，もし物理セクタ長が256バイトだったとすると，4物理セクタが論理セクタを表すことになります．論理セクタから物理セクタへの展開もドライバの役割です．

●クラスタ

もう一つは，ファイルの管理がすべてクラスタという単位で行われるということです．クラスタとは，OSがファイルを扱う際の最小の管理単位のことで，ディスクへのリード/ライト動作はこのクラスタ単位で行われます．ただし，デバイス・ドライバからみると，DOSからのリード/ライトの要求はすべて論理セクタの単位で指定されますので，クラスタの存在を意識する必要はありません．

普通1クラスタに対して複数の論理セクタが割り当てられていて，たとえ1バイトのファイルであっても1クラスタが使われます．すなわち，1クラスタが4論理セクタからなっていたとすれば，実際には使用されない残りのセクタ(第2～第4論理セクタ)は，他のファイルからは使うことができません．この物理セクタ，論理セクタ，クラスタの関係をまとめると図1のようになります．

このクラスタの大きさは，物理セクタの2のべき乗になるようにします．MS-DOSでは，1クラスタの大きさを，フロッピ・ディスクの場合で512～1024バイト，ハード・ディスクの場合で4096～16384バイト程度にするのが普通です．ところが，OS-9では，フロッピでもハード・ディスクでも，クラスタの大きさを1論理セクタ長の256バイトで使用する場合がほとんどです．1クラスタ256バイトは，フロッピ・ディスクはまだしも，数10Mバイト・クラスのハード・ディスクに対しては，小さすぎる値です．

これは，OS-9の標準コマンドであるフォーマット・コマンドのデフォルト値が1クラスタを1論理セクタにしているためです．OS-9の場合，ディスクの種類に関係なく，すべて一つのフォーマット・コマンドでフォーマットできます．フロッピでも，600Mバイトのハード・ディスクでもまったく同じです．

多くのOS-9ユーザは，大容量のハード・ディスクでも，フォーマット・コマンドのデフォルト値のままで使っている場合が多いのではないかと思います．OS-9のフォーマット・コマンドは，パラメータでクラスタ・サイズを自由に設定できます．少なくとも，40Mバイト以上のハード・ディスクに関しては，1クラスタを4論理セクタ(256×4＝1024)程度にしたほうがアクセス速度などの点からも有利と思われます．

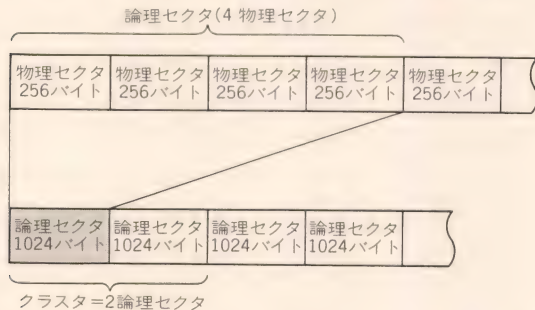
ちなみに，MS-DOSは，デバイス・ドライバとの間にフォーマットに関する明確な定義がないため，新しくハード・ディスクのドライバを作ったら，合わせて

〔図1〕物理セクタ，論理セクタ，クラスタの関係

論理セクタ(4物理セクタ)

物理セクタ 256バイト	物理セクタ 256バイト	物理セクタ 256バイト	物理セクタ 256バイト	物理セクタ 256バイト	

論理セクタ 1024バイト	論理セクタ 1024バイト	論理セクタ 1024バイト	論理セクタ 1024バイト	

クラスタ＝2論理セクタ

- この図は，物理セクタが256バイトのディスクに対して，論理セクタを1024バイト，1クラスタを2048バイトにした例．
- 物理セクタは，デバイス・ドライバがディスクに対してハード的にリード/ライトする単位．
- 論理セクタは，OSがデバイス・ドライバに対してリード/ライトを要求する単位．
- クラスタはOSがファイルを管理する最小の単位．

専用のフォーマット・コマンドも作らなければなりません．

●ディスクの最大容量

つぎに，OSの扱える最大のファイル容量ですが，MS-DOSは，67Mバイトで最近ようやく134Mバイトになりました．この制限は単純で，MS-DOSが扱う論理セクタ番号が16ビットであり，0～65,535しか表せないためです．そのため，ディスクの容量を増やすには論理セクタ長を大きくする以外方法はありません．たとえば，1論理セクタ長を1024にした場合，1,024×65,536＝67,108,864(67Mバイト)，2048にした場合で，2,048×65,536＝134,217,728(134Mバイト)が最大です．

一方，OS-9は8ビットの時代から4.2Gバイトまで扱えるようになっています．これは，この論理セクタ番号が最初から24ビット(2^{24}＝16,777,216)あるためで，1論理セクタが256バイトであっても256×16,777,216＝4,294,967,296(4.2Gバイト)まで可能です．最近，大容量のSCSIディスクが安価に入手できるようになりましたが，最初から24ビットにしてある先見性には驚かされます．

●ディレクトリ

どちらも，1ファイル当たりで使用するバイト数は32バイトで同じですが，その内容は異なります．両者で共通するのは，ファイル名とそのファイルがある場所を指す位置情報です．ただし，この位置情報は，MS-DOSの場合はクラスタ番号で，OS-9の場合は論理セクタ番号(LSN)で表します．

MS-DOSのディレクトリは，図2(a)のような構造に

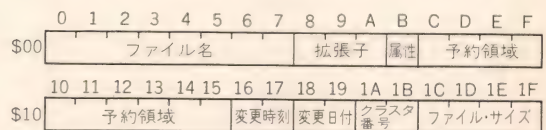
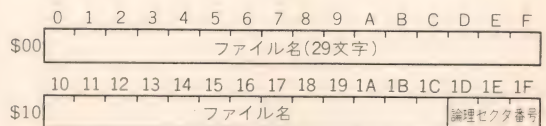
〔図2〕MS-DOSとOS-9のディレクトリ構造

	0～7	8～A	B	C～F
$00	ファイル名	拡張子	属性	予約領域

	10～15	16～17	18～19	1A～1B	1C～1F
$10	予約領域	変更時刻	変更日付	クラスタ番号	ファイル・サイズ

(a) MS-DOSのディレクトリ構造

	0～F
$00	ファイル名(29文字)

	10～1C	1D～1F
$10	ファイル名	論理セクタ番号

(b) OS-9のディレクトリ構造

なっていて，ファイル名，クラスタ番号のほかにそのファイルの情報(変更日付，属性，ファイル・サイズ)もすべてディレクトリ上にあります．そのため，ファイル名が8文字＋拡張子3文字までという制限があるものの，ディレクトリを読むだけでファイルの情報を得ることができ，それだけアクセス速度の点で有利です．

一方，OS-9のディレクトリは，図2(b)のようにファイル名とそのファイルの論理セクタ番号しかありません．その代わり，ファイル名が29文字まで使え，ファイル名を付けるのに悩まされる必要がありません．また，小文字も扱えるため見やすくなります(MS-DOSではすべて大文字)．ただ，OS-9の世界では，慣例的にディレクトリを大文字で，ファイル名を小文字で表すようにしています．

それでは，OS-9のファイル情報はどこにあるかというと，各ファイルごとに存在するファイル・デスクリプタという特別のセクタ内にあります．OS-9では，ファイルの詳細情報を，ディレクトリや実際のファイルとは別に1セクタを割り当てています．ディレクトリ内にあるファイルの位置情報は，じつはこのファイル・デスクリプタのある位置を指しています．ファイル・デスクリプタについては，後で詳しく説明します．

〔図3〕IDセクタとBPB

3バイト	メディアの総セクタ数
1バイト	1トラック当たりのセクタ数
2バイト	アロケーション・ビットマップのバイト数
2バイト	1クラスタ当たりのセクタ数
3バイト	ルート・ディレクトリの開始セクタ
2バイト	オーナID
1バイト	ディスクの属性
2バイト	ディスクのID
1バイト	ディスクのフォーマット(密度など)
2バイト	1トラック当たりのセクタ数
2バイト	予備
3バイト	ブート・ファイルの開始セクタ
2バイト	ブート・ファイルのサイズ
5バイト	作成日時
3バイト	ボリューム名
以下オプション領域	

(a) OS-9のIDセクタ

2バイト	1セクタ当たりのバイト数
1バイト	1クラスタ当たりのセクタ数
2バイト	予約セクタ数
1バイト	FATの数
2バイト	ルート・ディレクトリの登録数
2バイト	メディアの総セクタ数
1バイト	FAT ID
2バイト	1FAT当たりのセクタ数

(b) MS-DOSのBPB

● IDセクタ

IDセクタという呼び名は，OS-9で使っている言葉です．OS-9では，ディスクの一番最初の論理セクタ0をIDセクタとして明確に定義しています．そのため，いかなる種類のディスクでも，論理セクタ0を読み出すことで，そのディスクに関する情報を得ることができます．図3(a)は，IDセクタの構造です．OSあるいはデバイス・ドライバは，これらの情報をもとにしてファイルの管理を行います．

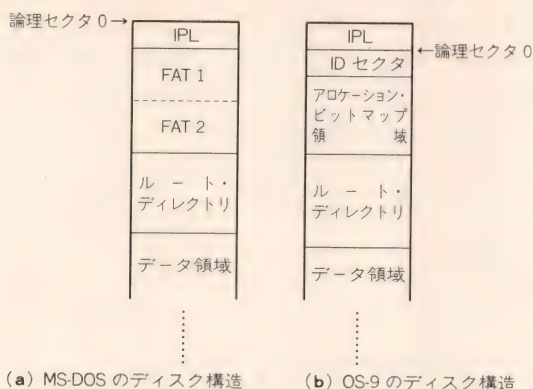
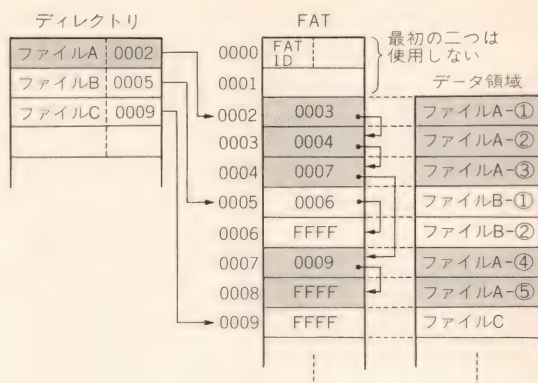
MS-DOSの場合，このIDセクタという概念はなく，これに代わるものとして図3(b)に示すBPB(BIOS Parameter Block)情報というものがあります．ところが，このBPB情報は，デバイス・ドライバに委ねられていて，必ずしもディスク上に存在するとはかぎりません．デバイス・ドライバによっては，ドライバ自身の中に定数としてもっていたり，あるいはディスクの一部にBPB情報を書き込んでおき，起動時に読み込む場合もあります．ディスクから読むといっても，MS-DOSとして，明確にBPB情報を書き込んでおく場所を定めていないため，非常にやっかいです．ただ，フロッピ・ディスクに関しては，一応ディスクの先頭にあるIPL部分にBPB情報をもつようになっています．

しかし，ハード・ディスクに関しては，いっさい取り決めがなく，各メーカごとに異なっています．そのため，同じインターフェースのハード・ディスクを違うメーカのMS-DOSパソコンに接続しても動作しないわけです．

● MS-DOSのファイル管理

図4(a)に，MS-DOSのディスク構造を示します．最初の何セクタかは，IPLとして使われDOSの管理外にあります．管理外といっても，論理セクタ番号はIPL部分から始まります．IPLとして使用するセクタは，BPB情報の中の予約セクタ数として定義されていま

〔図4〕MS-DOSとOS-9のディスク構造

（a）MS-DOSのディスク構造　（b）OS-9のディスク構造

〔図5〕MS-DOSのファイル構造

す．DOSは，最初のセクタからこの予約セクタとなっている部分を，ファイル管理の対象から除外します．また，ディスクによっては，このIPLは存在しない場合もあります．たとえば，RAMディスクなどがそうで，この場合FATの先頭が論理セクタ0となります．もちろん，このときBPBの中の予約セクタ数も0となります．

実際にDOSが管理するのは，FAT領域からです．このFATはまったく同じものが二つ連続して存在します．FATのあとに，ルート・ディレクトリ領域，そして実際のデータ領域が始まります．

MS-DOSにおけるファイル管理の中心となるのが，FAT(File Allocation Table)です．FAT自身はたんなるテーブルで，そのテーブル番号が全ディスク上のクラスタと1対1で対応しています．したがって，FATのエントリ数は，そのディスク全体のクラスタの総数になります．MS-DOSの場合，このクラスタ番号を12ビットで表す場合と16ビットで表す場合で，それぞれ12ビットFAT，16ビットFATといいます．

初期のMS-DOSは，現在のような大容量ディスクを想定しておらず，すべて12ビットFATで管理していました．12ビットFATは，容量の小さいディスクに対しては，ディスク全体でFAT部分の占める割合が少なくて済むという以外は，不便この上ありません．12ビットいう中途半端な値のため3バイトで2クラスタを表すことになり，直感的にクラスタ番号を知ることが非常に困難です．しかも，CPUがインテル系のため，さらに上位と下位が入れ替わっています．もちろん，これらのFATの内容は，通常のアプリケーション・レベルから意識する必要はありませんが，自分でディスク・ドライバやフォーマット・プログラムを書くような場合は，知っておかなければなりません．

一方，16ビットFATの場合，この面倒なことはありません．FATを2バイトごとにみていけば，簡単にクラスタ番号を求めることができます．もちろん，上位・下位が入れ替わっていることは同じです．以下，ここでは，話を簡単にするため，16ビットFATを例に説明をします．

16ビットFATの場合，クラスタ番号は0～$FFFF(65,535)まで表すことができます．しかし，このうち，0と1および$FFF7から$FFFFは特別な意味をもち，クラスタ番号としては使用しません．0は未使用のクラスタを表し，$FFFFは各ファイルの最後のクラスタを表すのに使用されています．また，$FFF7は欠陥クラスタを表します．交替処理のないディスクで，もしエラーのあるセクタがあったら対応するクラスタのFATを$FFF7にすることで管理の対象から外されます．

さて，このFATとクラスタの関係ですが，基本的には，**図5**のようになります．FAT領域の最初の2個は使用しません．ただし，FATの一番最初の1バイトは，FAT IDとして，フロッピ・ディスクのように外見は同じでもフォーマットが異なるメディアの識別に使用しています．

実際のクラスタ番号は0002から始まり，ここがディスク上のデータ領域の先頭を示します．データが複数のクラスタにまたがる場合，クラスタ番号がつぎのクラスタ番号を指すことでチェーンしていきます．そしてチェーンの最後のクラスタの場合は，$FFFFにな

っています．データが1クラスタ内で収まる場合は，つぎのクラスタが存在しないわけですから，$FFFFになります．**図5**のファイルCがそうです．

このように，MS-DOSの場合，FATのチェーンによってファイルを連結しています．そのため，万一FATの一部が破壊されると，ディスク全体が使用できなくなります．FATが2個あるのはこのためで，最初のFATがダメージを受けると2番目のFATを使用するようになっています．しかし，この方法は，信頼性の点からみるとあまり有効とはいえません．なぜなら，通常このFATは同一のトラック上に存在するため，トラブルがあったときは同時破壊される可能性が高いからです．

● OS-9のファイル管理

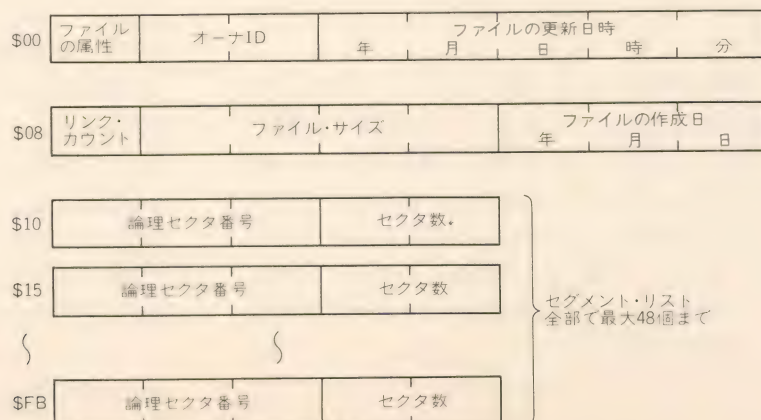
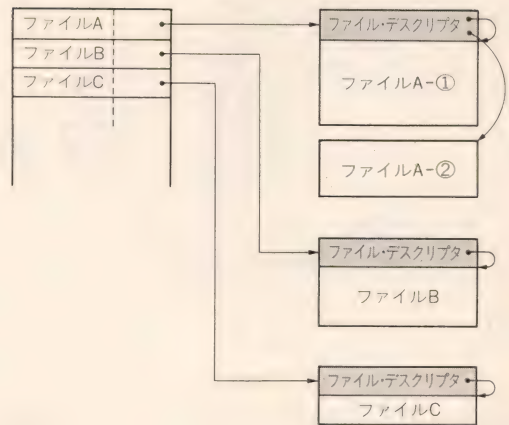
一方，OS-9の管理方法はMS-DOSとはまったく異なります．**図4**(b)にOS-9のディスクの構造を示します．最初のIPL部分はMS-DOS同様OSの管理外の領域です．OS-9の管理領域はIDセクタから始まり，ここが論理セクタの0となります．OS-9の場合，MS-DOSと違って，IPL領域の有無にかかわらず，IDセクタのある位置が論理セクタ0となります．

そして，IDセクタの後に，OS-9独特のアロケーション・ビットマップ領域があります．アロケーション・ビットマップとは，ディスク上の全クラスタを1ビット単位で，使用しているか未使用かを表します．これは，マルチタスクで動作するOS-9にとっては重要な情報です．複数のタスクから同時使用されることがあるOS-9のディスクは，このビットマップ情報によって，各セクタの排他制御を行います．また，ディスクのフォーマット時に欠陥セクタが発見された場合も，対応するクラスタのビットがセットされます．

アロケーション・ビットマップ領域の後は，MS-DOSと同じように，ルート・ディレクトリ領域，そして実際のファイル領域が始まります．

OS-9は，このアロケーション・ビットマップとセグメント・リストというものでファイルを管理しています．セグメント・リストとは，MS-DOSのFATの役割をするもので，このリストによってファイルを連結します．OS-9は，ディスクにデータを書き込むとき，なるべく連続したセクタに書き込もうとします．しかし，連続した空き領域が確保できない場合，空いている領域を捜しながら，データを分割して書込みを行います．この分割した単位をセグメントといい，この情報のリストをセグメント・リストといいます．

セグメント・リストは，先にふれたファイル・デスクリプタ内にあります．ファイル・デスクリプタは，一つのファイルあるいはディレクトリごとに一つ存在します．したがって，OS-9の場合，どんなに小さなファイルでも最低2セクタを使用することになります．実際，ファイル・サイズが0の場合でも，ディレクトリ上にファイル名が存在するかぎりファイル・デスクリプタが存在します．

図6にこのファイル・デスクリプタの構造を示します．最初の16バイトをファイルの情報として，残りの240バイトをセグメント・リストとして使用しています．セグメント・リストは，5バイトで構成され，最初の3バイトがファイルの開始論理セクタ番号，つづく2バイトが連続している論理セクタの数を意味します．つまり，ある論理セクタ位置から何セクタが連続

〔図6〕
ファイル・デスクリプタの構造

しているかを表すリストです．フォーマット直後のディスクはセクタが連続しているため，このリストは1個で(あるいは2個程度で)すみます．しかし，長い間ファイルの作成，削除を繰り返し，ディスク内のセクタが分断されてくると，一つのファイルがディスク上の空き領域に点在することになります．このとき，点在している数だけセグメント・リストが作られます．

ところが，このセグメント・リストの数は48(240÷5)個ぶんしかありません．そのため，あまりにもファイルが分断されると，この数だけではファイルの連結ができなくなり，セグメント・リスト・フルのエラーになってしまいます．このエラーは，OS-9のファイル管理における唯一の欠点(逆に長所でもあるが)ともいえるものです．これを防ぐために，OS-9にはファイルを新たに作るときあらかじめファイルの領域を確保する機能があります．OS-9のCopyコマンドは，ファイルをコピーするとき，あらかじめコピー元のファイル・サイズを調べ，コピー先のファイル領域を確保するようになっています．

図7にOS-9のファイル構造を示します．この図ではファイル・デスクリプタと実際のファイルが連続しているようになっていますが，これは必ずしも連続するとはかぎりません．ファイルの分断のぐあいによってはまったく別々の場所に確保されます．このようにOS-9では，セグメント・リストがファイル・デスクリプタという1セクタの中にしかないという欠点はあるものの，この重要な情報を各ファイルごとにもっているため，逆にいうとファイルが壊されにくいという長所にもなっています．

MS-DOSの場合，FATが破壊されると，そのディスクに含まれる全ファイルを失うことになります．しかし，OS-9の場合，仮にディスクの最初にあるアロケーション・ビットマップ情報やルート・ディレクトリを万一破壊しても，ほとんどの場合，ファイルを修復することができます．これは，それぞれのファイルの管理情報がディスク内に点在しているためです．ディスクの先頭から全セクタを調べ，特徴的なパターンであるファイル・デスクリプタが発見できれば，ファイルの実体は確実に修復できます．

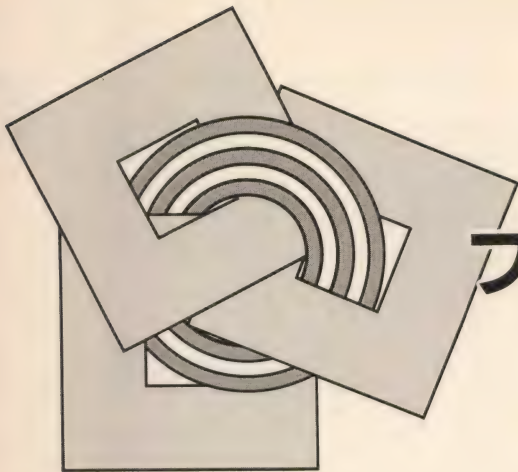
また，ディスク上の各ファイルは，1回のリード/ライト動作で完結した状態になっています．そのため，書込みを行っているときにディスクを抜いたり，あるいは停電などがあっても，ディスクを壊す可能性がきわめて低くなっています．最悪の場合でも，書込みを行っているファイルが影響される程度ですみます．

〔図7〕OS-9のファイル構造

おわりに

今回は，ファイル管理というより，ディスクの構造を中心に話を進めてきました．筆者は，長年OS-9に携わってきましたが，最近，仕事の関係でMS-DOSのファイル構造を学ぶ機会を得ました．そこで，今回の特集に合わせて，MS-DOSとOS-9を比較してみました．ただ，ここで，両者を比較してどちらが優れているかなどと論ずるつもりはありません．MS-DOSを知っている方がOS-9のことを，逆にOS-9のことを知っている方がMS-DOSのファイル構造について少しでも知っていただければ幸いです．

なおOS-9では，その柔軟な構造から，ファイル・マネージャのレベルで，MS-DOSのディスクを完全にアクセスできるようになります．この，MS-DOSのディスクを扱うためのファイル・マネージャはすでに発表されています．

ほし・みつゆき　㈱星光電子

ファイル・マネージャの構造と設計

菅原 宏和

OS-9/68Kの最大の特徴の一つは，ファイルの論理的な取扱いをファイル・マネージャというカーネルとは独立したモジュールに行わせていることである．したがって，時代的に物理的な装置が変化するのに応じて，論理的なファイルの取扱いも柔軟に変更させることができる．ここでは，デバイス・スタティック・ストレージ，パス・デスクリプタ，デバイス・テーブルなどのファイル・マネージャをとりまくデータ構造を説明した後，カーネルおよびデバイス・ドライバとのインターフェースについて解説する．　(編集部)

組込みシステムからパーソナル・コンピュータ，さらに小規模データ・プロセシングまでの幅広い範囲をカバーする本格的なOSとして，OS-9/68Kには数多くの特徴があります(以下OS-9とはOS-9/68Kのこと)．その中の一つに「モジュール構造」があります．OS-9以外のOSでも，モジュール構造を謳い文句にしているものがないわけではありませんが，その多くは，たんに概念上またはOS内部の構造がそうなっているというだけで，OS-9のように物理的に実際に分割できる単位としてのモジュールは，あまり見当たりません．とくに，OS-9では，OS自身でさえモジュールの組合せで構成されており，ターゲットに応じて非常に柔軟に対応できます．

本稿では，その中のファイル・マネージャにスポットをあて，その役割・構造と実際の作成例について解説します．

I　ファイル・マネージャの基礎

1 ファイル・マネージャとは

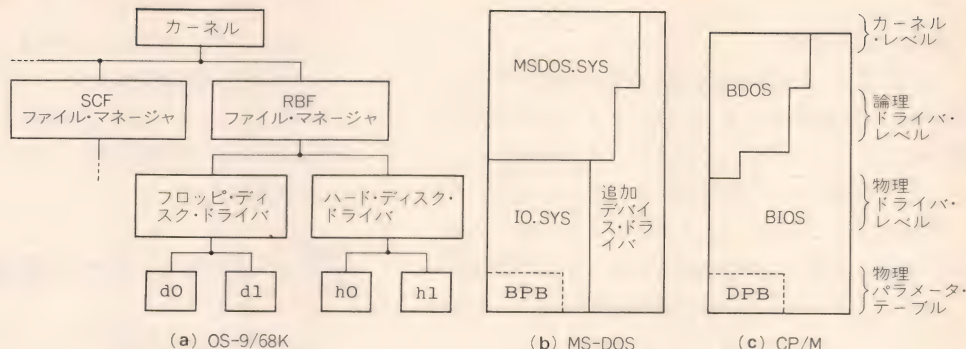
1.1　他の処理系との比較

図1は，毎度おなじみのOS-9の入出力関係のモジュール構造を，MS-DOSおよびCP/Mの入出力プログラム構造と比べたものです．これを見ると，OS-9に以下のような特徴があることがわかります．

① OS-9は，カーネル/ファイル・マネージャ/デバイス・ドライバ/デバイス・デスクリプタの各モジュールが完全に分離していて，システムの稼働中にも動的に追加・削除できるが，MS-DOSとCP/Mではシステムを再構成するか，少なくとも再起動しないと新規デバイスの追加・削除などができない．

② 論理ドライバ・レベルのプログラムが，OS-9では「ファイル・マネージャ」として独立しているが，MS-DOSではカーネル(MSDOS.SYS)と追加デバイス・ドライバに，CP/MではBDOSに組み込まれている．

CD-ROMや光ディスクなど，新しい論理構造をもつデバイスを追加するためには，OS-9ではハードウェアに依存しないファイル・マネージャと，ハードウェアに依存するデバイス・ドライバを用意します．このうち，そのファイル・デバイスの論理的操作だけを扱うファイル・マネージャは，ハードウェアに依存しないので，変更することなく他のハードウェアで使用でき

〔図1〕
入出力プログラム構造の比較

ます．MS-DOSでは，基本的にはブロック型デバイスとキャラクタ型デバイスの2通りしかありませんから，まったく新規の論理構造をもつデバイスを扱うためには，カーネルから作り直す(MS-DOS 2.x→3.x)か，またはハードウェアに依存する追加デバイス・ドライバで，むりやりブロック型またはキャラクタ型に整合させなければなりません．CP/Mでは，新しい論理構造をもつデバイスの出現などは予想していませんから，OS(というほどのシロモノではないが)レベルの新規デバイスの追加は不可能です．

1.2 ファイル・マネージャの役割

OS-9のファイル・マネージャの役割は，大きく分けて以下の二つです．

(1) ファイルの論理構造の操作

これは，他の多くの処理系で実施されているのと同じ発想です．物理的には異なるが，論理的に同じ構造をもつ多くのデバイスを一つのプログラム・モジュールで管理するためです．ただし，OS-9以外の原始的な処理系では，物理ドライバ・レベルとの区分けが不適切であったり，将来の見通しが貧弱なため，それが後で重い足枷となっている場合もあります．

たとえば，CP/Mでは，BDOS内にトラック/サイドのようなメディアに関する物理パラメータの処理を含んでいるため，複雑怪奇なDPB(ディスク・パラメータ・ブロック)で非生産的な処理を行っています．

MS-DOSでは，開発者がせいぜい1Mバイト程度の容量のフロッピ・ディスクしか想定していなかったため，FATエントリが表現できるクラスタ番号が小さすぎ，大容量ハード・ディスクでは，クラスタ・サイズを大きくしてディスク容量をむだにするか，互換性に目をつぶって「拡張FAT」を使用するかの選択を迫られます．

しかも，これらの処理系では，論理ドライバ部分が，OS本体すなわちカーネル部と一体化されているため，変更はメーカ側で行わなければならず，ユーザ側で追加・変更することはできないのが普通です(注)．

(2) 排他制御

排他制御(相互排除処理ともいう)は，MS-DOSやCP/Mのようなシングル・タスクのディスク・ユーティリティしか使ったことのない人にとっては，あまり縁がないかもしれませんが，OS-9のようなマルチタスクのOSでは重要な問題です．つまり，OS-9では，一般的には複数のプログラム(プロセス)が同時に走行しており，同一の物理デバイスに対して複数のプロセスからの使用要求が同時に発生することがあります．

たとえば，ディスクに書込みを行うときには，新しい未使用セクタをビット・マップから確保する必要がありますが，二つ以上のプロセスが同一ディスクに書込みを行うときには，何らかの排他制御をしないと，別のファイルが同じセクタを使用してしまう危険があります．この排他制御は，ファイル・マネージャでなくても，①アプリケーション・プログラム，②カーネル，③デバイス・ドライバのいずれの段階でも不可能ではありませんが，これらのレベルで排他制御を行うことは，以下のような理由で適切とはいえません．

① アプリケーション・プログラムの場合

MS-DOSのプリント・スプーラの例でもわかるように，アプリケーション・プログラムに複雑な排他制御処理を組み込むことは，個々のプログラムに負担がかかり，一般性・互換性はなく，特定の状況でしか通用

(注) MS-DOSのつぎのリリースで実現される予定のIFS(Installable File System)では，この問題は改善されます．

しません.

② カーネルの場合

カーネルは，一部の入出力に関する排他制御を行いますが，その範囲はかぎられています．排他制御が必要かどうかは，そのファイル・デバイスの特性によります．カーネル・レベルでは，すべてのファイル・デバイスの見かけは同じで，そのファイル・デバイスに対する操作が本当に排他処理を必要とするのかはわかりません．カーネルで一律に排他制御を行うとシステムのパフォーマンスが低下し，シングル・タスクに近い動作になってしまうおそれがあります.

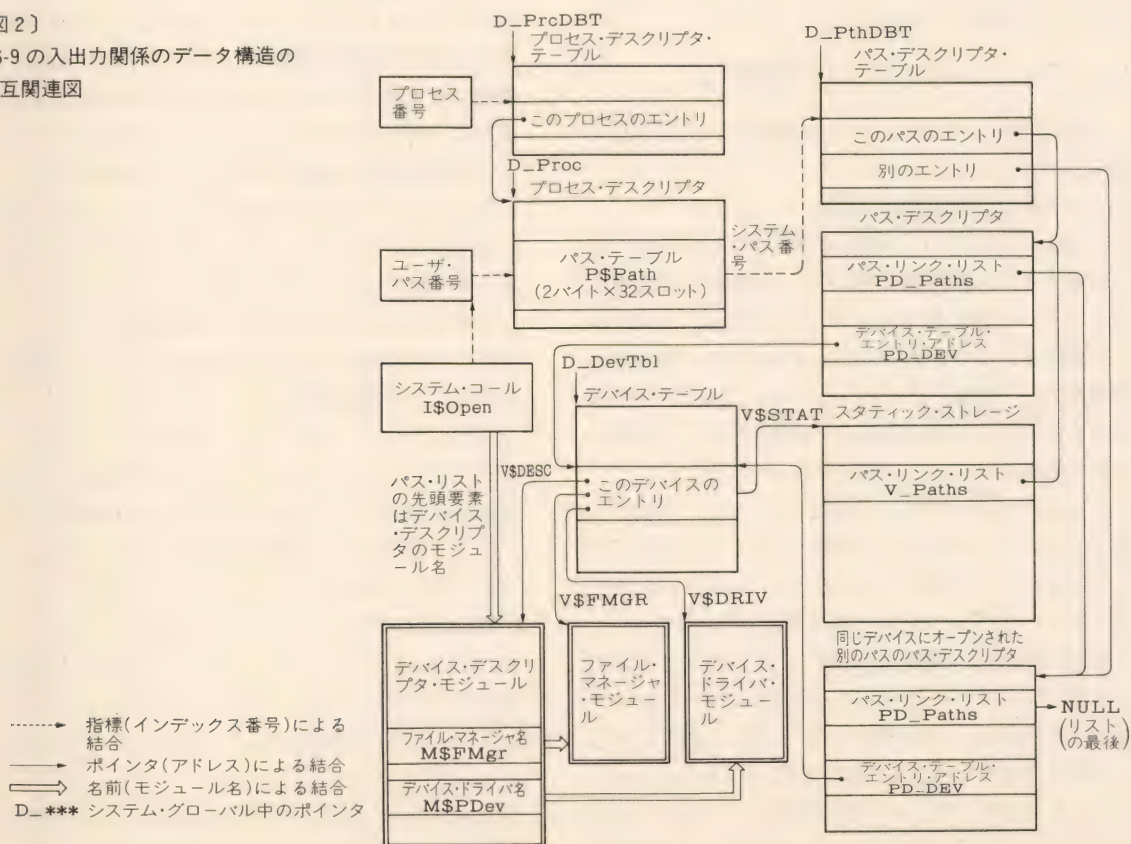
③ デバイス・ドライバの場合

OS-9 のデバイス・ドライバは，操作内容をできるだけ単純化するように設計されています．たとえば，RBF(ディスク)ファイル・マネージャは，ディスクの読み書きを，「指定した論理セクタ番号(LSN)から指定したセクタ数を読み書きする」という単純な操作をデバイス・ドライバに指示するだけであり，それがビットマップ・セクタなのか，ディレクトリなのか，それともファイル本体なのかはデバイス・ドライバはわかりません．したがって，物理セクタの読み書きなどの基本的な排他制御を除けば，ビットマップのロックやレコード・ロックなどの RBF の管理するディスク構造に起因する排他制御をデバイス・ドライバで行うことは不経済です.

2 システム・データ構造

OS-9 の内部構造を理解するためには，いくつかの重要なシステム・データ構造を理解しなくてはなりません．ここでは，ファイル・マネージャすなわち入出力に直接関係する ① デバイス・スタティック・ストレージ，② パス・デスクリプタ，③ デバイス・テーブル，の三つの主要なシステム・データ構造について解説します.

OS-9 には，以上の三つ以外にも，間接的に入出力操作に関係するいくつかの重要なシステム・データ構造があります．図 2 に，I$Open システム・コールで，あ

〔図 2〕
OS-9 の入出力関係のデータ構造の相互関連図

るデバイスに対してパスをオープンしたときに返されるユーザ・パス番号から，それらのシステム・データ構造がどのように結合されるのかを示します．なお，**図2**中のデータの関係は，カーネルやファイル・マネージャだけが関与し，ユーザ・プログラムには関係ありません．

2.1 デバイス・スタティック・ストレージ

デバイスのハードウェアに1対1に対応したデータを保持し，デバイスごとにただ一つ存在します．スタティック・ストレージ中には，そのデバイス・ポートのベース・アドレス，現在のハードウェアの状態，割込みのマスク・ワード，現在そのデバイスを使用しているプロセスのプロセスID，デバイスが割込みを発生するのを待っているプロセスのプロセスIDなど，そのデバイスの静的な情報が記録されています．

▶デバイス・スタティック・ストレージの構造

スタティック・ストレージは，**図3**に示すように，① 全OS-9デバイスに共通な部分，② ファイル・マネージャごとに共通な部分，③ デバイスごとに異なる部分の三つの部分に分けられます．

(1) 全デバイスの共通部分(オフセット：$00～$2d)

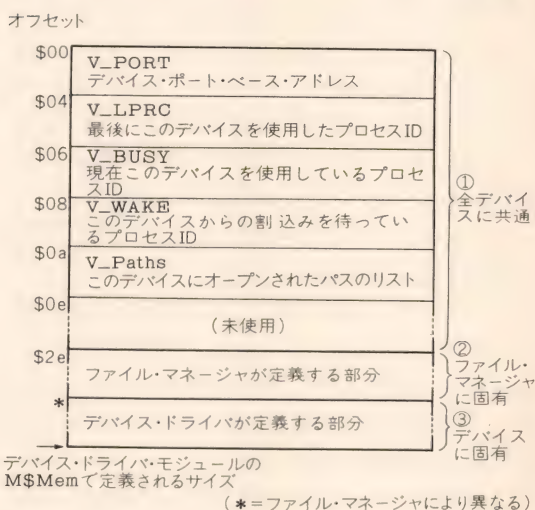
主としてカーネルが設定します．ファイル・マネージャはV_BUSYを使用してデバイスの相互排除処理を行います．デバイス・ドライバは，V_PORTからハードウェア・ポート・アドレスを得ることと，V_BUSY，V_WAKEを使用してハードウェアの動作の完了を待つための待機と再起動のために使用します．

(2) ファイル・マネージャの固有部分(オフセット：$2e～ファイル・マネージャの定義する値)

カーネルは，この部分のメモリを確保して，オール・ゼロに初期化する以外，いっさい関知しません．メディアの総サイズやファイル・デバイス固有の相互排除処理のためのリストなど，そのファイル・デバイスに固有の論理的な静的データが記憶されます．

(3) デバイス・ドライバの固有部分(オフセット：ファイル・マネージャの定義する値～デバイス・ドライバの定義する値)

デバイス・ドライバだけが使用します．カーネルやファイル・マネージャは関知しません．割込みマスク・ワードやDMACのアドレス，サーキュラ・バッファのポインタなど，ハードウェア(デバイス・ドライバ)に

〔図3〕デバイス・スタティック・ストレージの構造

固有の静的データが記憶されます．(1)，(2)と合わせてデバイス・ドライバのvsectに宣言します．リンカは，合計サイズをデバイス・ドライバのモジュール・ヘッダのM$Memフィールドに書き込みます．

▶デバイス・スタティック・ストレージの確保と解放

スタティック・ストレージは，そのデバイスがI$Attachシステム・コールによりシステムに登録されるときに，デバイス・ドライバのモジュール・ヘッダのM$Memフィールドに書かれたサイズのメモリをカーネルが確保し，ごく一部を初期化(全バイトをゼロにしてV_PORTを設定)した後，デバイス・ドライバにより初期化されます．

ユーザが明示的にI$Attachシステム・コールを発行しなくても，I$Openなどでそのデバイスにパスをオープンしようとしたときに，カーネルは内部でI$Attachを実行します．I$Attachが実行されても，そのデバイスがすでに登録されていれば，再び登録・初期化するようなことはなく，たんにそのデバイスのユース・カウント(デバイス・テーブル内のV$USRS)を増やすだけです．

MS-DOSなどでは，そのデバイス・ドライバ自身のコードの一部を作業領域として使用しますが，OS-9ではモジュール自身を書き換えるようなプログラム，すなわち自己修正コードは禁止されていますから，作業領域(スタティック・ストレージ)は，デバイス・ドライバ・モジュールとは別個に確保されたメモリ領域で

す．また，このようにデバイス・ドライバとデータ領域は完全に分離していますから，デバイス・ポート・ベース・アドレスが異なるデバイス・デスクリプタをもってくれば，一つのデバイス・ドライバで複数の物理デバイスを操作できます(次項を参照)．この意味では，他のほとんどのOS-9のプログラムと同様に，デバイス・ドライバはリエントラント(再入可能)です．

スタティック・ストレージは，そのデバイスがもはや不要になり，I$Detachシステム・コールによりシステムから削除されるときに，カーネルにより削除されます．スタティック・ストレージの削除に先立ち，そのデバイスはデバイス・ドライバにより終了の処理(割込みの禁止やバッファの切離しなど)がなされます．I$Detachは，I$Closeシステム・コールでパスがクローズされるときにも内部で実行されますが，I$Attachが増加したユース・カウントを減少させ，そのデバイスのユース・カウント(V$USRS)が0にならないと，デバイスの削除は実行されません．

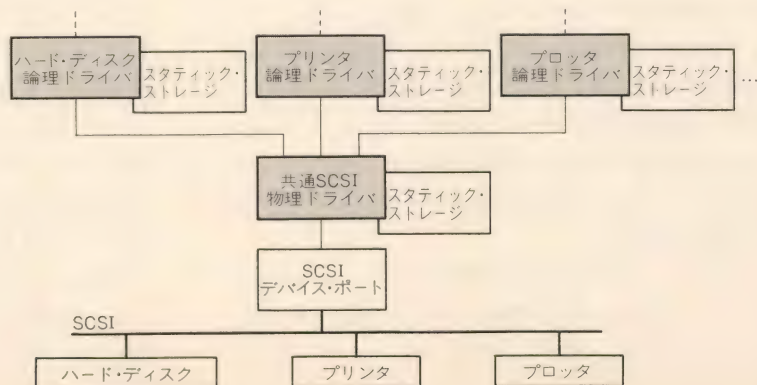
▶デバイス・ポート・ベース・アドレス(M$Port)

冒頭に「デバイス・スタティック・ストレージはデバイスのハードウェアに1対1に対応したデータを保持する」と書きましたが，じつは「デバイスのハードウェア」という表現は正確ではありません．なぜならば，RAMディスクやパイプのように，実際のハードウェアが存在しないデバイスもあるからです．カーネルは，別々のデバイス・デスクリプタに書かれたデバイス・ポート・ベース・アドレス(M$Port)とデバイス・ドライバのモジュール名(M$PDev)が等しければ，同一デバイスと解釈します．

注目すべきことは，ポート・ベース・アドレスが同じでも，デバイス・ドライバが違えば，別のデバイスとして扱われ，独立したスタティック・ストレージを割り当てられるということです．たとえば，RAMディスク(/r0)とパイプ(/pipe)は，両方とも(標準のデバイス・デスクリプタでは)ポート・ベース・アドレスは$0ですが，デバイス・ドライバが違いますから，別々のデバイスです．ところが，/pipeと/nilは，両方ともポート・ベース・アドレスは$0で，デバイス・ドライバ・モジュールがnullですから，カーネルはこれらを同一デバイスとして扱い，共通のスタティック・ストレージを一つだけ割り当てます．したがって，実際の物理デバイス・ポートがない仮想デバイスを追加するときには，このことに注意しないと，スタティック・ストレージが衝突してシステムがクラッシュすることがあります．

もしダミー・ドライバとしてnullを使いたければ，/nilや/pipeがすでに使用している$0以外のデバイス・ポート・アドレス(たとえば$1)を使用します．/nilと/pipeは同一のスタティック・ストレージを使用しますが，互いに衝突しないようになっています．

このことを積極的に使用して，同一の物理デバイス・ポート(たとえばSCSI)に接続された複数の異なるファイル・マネージャに属するタイプのデバイス(たとえばハード・ディスク(RBF)とテープ(SBF))を単一のデバイス・ドライバで制御することもできそうですが，スタティック・ストレージの定義が各ファイル・マネージャごとに異なりますので，実際には困難でしょう．マイクロウェアから最近発表された「2レベル・デバイス・ドライバ」では，同一SCSIポートで制御される各デバイス(ハード・ディスク，プリンタ，プロッタなど)ごとに別々の論理デバイス・ドライバがあり，

〔図4〕
2レベル・デバイス・ドライバ

それらの論理デバイス・ドライバが共通の物理デバイス・ドライバを呼び出すという構造になっているようです(図4)．

2.2　パス・デスクリプタ

デバイス・スタティック・ストレージがシステムにアタッチされたデバイス・ポートごとに存在するのに対して，パス・デスクリプタはオープンされたパスごとに存在します．

パス・デスクリプタは，あるパスが I$Open などのシステム・コールでオープンされるときにカーネルにより確保され，一部分が初期化された後，ファイル・マネージャの Open(または Create)のエントリでそのファイル・デバイス固有の初期化がなされます．パス・デスクリプタは，パスというファイルに対する論理的な経路と対応することからも想像できるように，主としてファイル・マネージャが使用します．

パス・デスクリプタは，図5のように四つの部分に分けられます．

(1) 全デバイスの共通部分(オフセット：$00～$29)

主としてカーネルが設定し，ファイル・マネージャとデバイス・ドライバが参照します．

PD_CNT(と PD_COUNT)は，パスがオープンされたときは1に設定され，I$Dup システム・コールでパスが複製されるたびに1ずつ増やされます．I$Dup は，たとえば Shell が標準入出力のリダイレクションを行うときに発行されますが，I$Fork で新しいプロセスがフォークされるときに親プロセスから引き継がれるパス(通常，標準入出力と標準エラー)が複製されるときにも，カーネル内部で実行されます．

子プロセスが F$Exit で終了するときには，そのプロセスでオープンしているすべてのパスが，F$Exit 内部で実行される I$Close によりクローズされますが，このときに PD_CNT(と PD_COUNT)は1ずつ減少され，つじつまが合うようになっています．

また，通常，標準パスは(リダイレクションされていなければ)同一のデバイス(ターミナルなど)に接続されていますが，じつはこれらの三つ(標準入力，標準出力，標準エラー)は別々のパスではなく，一つだけオープンされたパスが I$Dup で複製されたものです．このオープンと複製は，システム起動時にはカーネルが，TSS によるマルチユーザ動作時には TSMon(TSS モニタ)が行います．

〔図5〕パス・デスクリプタの構造

(2) ファイル・マネージャが定義する部分 ①
(オフセット：$2a～$7f)

オープンされたファイルの(論理的な)サイズ，位置，メディア上の位置，ブロック・バッファへのポインタとその内容など，パスのあらゆる論理的かつ動的な情報が書かれます．たとえば，RBF の SS_Pos の I$GetStt は，たんにこの領域にある PD_CP フィールドの値を返すだけです．この領域は，ファイル・マネージャだけが使用し，カーネルやデバイス・ドライバは使用しません．

(3) デバイス・デスクリプタからコピーされる部分
(オフセット：$80～　)

デバイス・タイプを表す先頭(オフセット $80)の

PD_DTP(M$DTyp)を除き，ディスクの総シリンダ数，面数，対話的入力時の後退・行複製・行削除文字など，デバイスごとに異なるパラメータで，デバイス・デスクリプタの初期化テーブル(M$DTyp~)からM$Opt で指定されるバイト数だけ，カーネルがパスのオープン時にコピーします．

したがって，デバイス・デスクリプタとは，見方を変えれば，パス・デスクリプタの初期化データ・テーブルともいえます．この領域は，ユーザ・プログラムから SS_Opt の I$GetStt で読み出すことができ，部分的に(ファイル・マネージャが許すフィールドだけ) SS_Opt の I$SetStt で変更できます．そのため，この領域は「パス・デスクリプタのオプション部分」とも呼ばれます．

(4) ファイル・マネージャが定義する部分 ②

(オフセット：　~$ff)

ファイル・マネージャだけが使用します．オフセット $2a~$7f の領域との違いは，この領域が SS_Opt の I$GetStt でオプション領域といっしょにユーザ・プログラムから読み出せることです．RBF と Pipe では，オフセット $e0~$ff(PD_NAME)の 32 バイトにそのファイルのファイル名が書かれています(パス・リストの最後の要素)．Pipe ファイル・マネージャは，PD_NAME の内容で，名前付きパイプと通常の(名前なし)パイプを区別します．

2.3 デバイス・テーブル

デバイス・テーブルは，スタティック・ストレージやパス・デスクリプタのようにデバイス・ドライバがその内容を扱うことはありませんので，OS-9 を移植す

コラム1

OS-9 のシンボル定義

OS-9 のアセンブラでは，アセンブル時に未解決のシンボルは，自動的に外部参照とみなされ，リンク時に解決されます．このことは，分割アセンブルができるアセンブラなら当然のことですが，OS-9 のアセンブラでは，サブルーチンやジャンプ先のアドレスだけではなく，構造体中のフィールドのオフセットや定数なども外部参照シンボルとして扱えるという特徴があります．

したがって，複数のソース・コード・ファイルから共通的に参照されるデータ構造や定数は，1 箇所(1 ファイル)だけで do や equ 疑似命令で定義して，その他のソース・コード中ではシンボルで参照します．各シンボルの実際の値は，リンク時に解決されます．この方法は，use により定義ソース・コード・ファイルをアセンブル時に取り込んで解決する方法(C の #include/#define と同じ方法)と結果的には同じようになりますが，以下の点でメリットがあります．

① アセンブルが高速(ソース・コード・ファイルを取り込まないぶんだけ高速)

② シンボルの解決をリンク時まで延期できる(とりあえず各ファイルをアセンブルして文法エラーなどを修正)

③ シンボルの値が変更されても，それを参照しているファイルをアセンブルしなおす必要はない(リンクのみやりなおし)

③をうまく利用すると，万が一システム定義の定数やオフセットが変更されたときにも，エンド・ユーザがリンクだけをやりなおすことにより，新しい定義に対応することができます．このような場合，通常はアセンブリ言語や高級言語でかかれたソース・コードのアセンブルやコンパイルからやりなおさなければなりませんが，商業的な理由により，エンド・ユーザにはソース・コードを公開できない場合もあります．OS-9 のように定数などのシンボルも外部参照できると，ユーザにはリロケータブル・オブジェクト・コードの形で提供することにより，ソース・コードの秘密保持ができます．

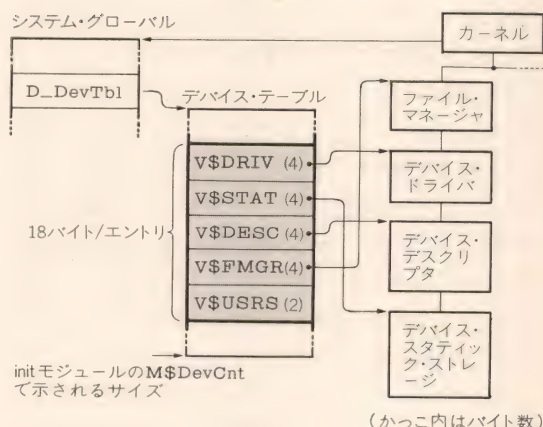
OS-9 のアセンブラと一緒に提供されるライブラリ・ファイル sys.l には，デバイス・ドライバやファイル・マネージャ・プログラムをアセンブルするときに必要な各種のシステム・データのシンボルが定義されています．**表A** に，これらのシンボルのネーミング・コンベンション(命名方法)による分類をしてみました．OS-9/6809 時代の名残で，$ がシンボル中にあるため，そのままでは C の #define による定数定義には使えないものもあります (OS-9/6809 時代の“.”は“_”になっている)．

余談ですが，MS-DOS の話題で必ず登場する INT 21H などという呼び方に虚しさを感じるのは，筆者にかぎらずすべての OS-9ers の共通の思いだと思います．まさか，まともなアセンブラ・プログラムさえなかった CP/M 以前の慣習が残っているというわけでもないでしょうが．OS-9 では，「F$Link のシステム・コールを発行したら E$MNF エラーが返された」といいますが，「機能コード 0 番の trap #0 を実行したら 221 番のエラーが返された」などとはいいません．

るために新しいデバイス・ドライバを開発するユーザにもあまり顧みられることがないのですが，OS-9 全体の入出力構造には大変重要なデータ構造です．なぜなら，デバイス・テーブルには，ファイル・マネージャ，デバイス・ドライバ，デバイス・デスクリプタ，そしてスタティック・ストレージという，あるデバイスを操作するために必要な階層的なモジュール間の関係および静的データ領域を結びつけるためのポインタが格納されているからです(図 6)．

▶デバイス・テーブルの作成

デバイス・テーブルは，システムの起動時にカーネルにより作成されます．カーネルは，init モジュールの M$DevCnt で示されるサイズ(×18 バイト)ぶんのメモリを確保し，内容をすべてゼロに初期化し，その先

〔図 6〕デバイス・テーブル

〔表 A〕システム定義シンボル

シンボル	用　法　と　例
B_***	複数のビットが意味をもつフラグ・バイト中のビット番号(LSB=0) 例：B_EofLck(RBF のレコード・ロック・ステータス)
C$***	文字コード 例：C$CR(キャリッジ・リターン)
D$***	デバイス・ドライバ・モジュールのエントリ・オフセット・テーブル中の該当エントリのオフセット 例：D$INIT(INIT ルーチンへのオフセット)
DD_***	ディスクの ID セクタのフィールド 例：DD_DIR(ルート・ディレクトリの LSN)
DT_***	デバイス・デスクリプタ内のデバイス・タイプ・コード 例：DT_RBF(RBF デバイス)
D_***	システム・グローバル内の変数 例：D_Proc(カレント・プロセス・デスクリプタへのポインタ)
E$***	エラー番号 例：E$BMode(不正なアクセス・モード)
Ev$***	イベント・セマフォ機能コード 例：Ev$Creat(新しいイベントの作成)
F$***	入出力以外のシステム・コール機能コード 例：F$Link(メモリ・モジュールのリンク)
FD_***	ファイル・デスクリプタ・セクタのフィールド 例：FD_DAT(ファイルの最終変更日時)
I$***	入出力システム・コール 例：I$Open(パスのオープン)
M$***	メモリ・モジュール・ヘッダのフィールド 例：M$Name(モジュール名へのオフセット)

シンボル	用　法　と　例
MD$***	モジュール・ディレクトリ・エントリのフィールド 例：MD$MPtr(モジュールへのポインタ)
P$***	プロセス・デスクリプタのフィールド 例：P$PID(親プロセスのプロセス ID)
PD_***	パス・デスクリプタのフィールド 例：PD_PD(システム・パス番号)
Q$***	IRQ ポーリング・テーブル・エントリのフィールド 例：Q$SERV(IRQ サービス・ルーチンの絶対エントリ・アドレス)
Q_***	プロセス・デスクリプタの P$QueuID フィールドの内容 例：Q_Sleep(スリープ状態)
R$**	ユーザ・レジスタ・スタック内のレジスタ 例：R$d0(ユーザから渡され，ユーザに返される d0 レジスタの内容)
S$***	シグナル・コード 例：S$Kill(プロセスを終了させるシグナル)
SS_***	I$GetStt/SetStt の機能コード 例：SS_Size(ファイル・サイズを設定または得る)
T$***	Math トラップ・ハンドラの機能コード 例：T$DAdd(倍精度実数の加算)
T_***	例外ベクタ・テーブルのエントリ 例：T_BusErr(バス・エラー・ベクタ)
V$***	デバイス・テーブル・エントリのフィールド 例：V$DESC(アタッチされているデバイスのデバイス・デスクリプタ・モジュールへのポインタ)
V_***	デバイス・スタティック・ストレージのフィールド 例：V_PORT(デバイス・ポート・アドレス)

頭アドレスをシステム・グローバル中のD_DevTblに記録します．デバイス・テーブルのサイズと場所は，システムが稼働しているかぎり，不変です．

I$Attachシステム・コールによりデバイスがアタッチされると，カーネルは，デバイス・テーブルの先頭から後方に向かって，指定されたデバイスのデバイス・デスクリプタがすでにデバイス・テーブルにアタッチされているかどうかを走査します．もしすでにそのデバイス・デスクリプタがアタッチされていれば，そのエントリのV$USRSをインクリメントするだけで，それ以上何もしません．

もしまだそのデバイス・デスクリプタがアタッチされていなければ，とりあえず空きデバイス・テーブル・エントリを確保して，ファイル・マネージャとデバイス・ドライバ(デバイス・デスクリプタ中にモジュール名が書かれている)をリンク(F$Link)してみます．もしリンクに失敗すれば，エラーで戻ります．

つぎに，物理的に同一だが論理名の違うデバイス(ポート・アドレスとデバイス・ドライバが同じデバイス)がアタッチされているかどうかを探し，もし見つかれば，V$DRIV～V$FMGRの内容をそのエントリから新しいエントリにコピーし，V$USRSを1に設定して終了します．

物理的に同一の別名デバイスも見つからなければ，まったく新規のアタッチですから，デバイス・ドライバのM$Memに書かれたサイズのスタティック・ストレージをシステム・メモリ・プールから確保して，内容を初期化して(全バイトにゼロを書き込み，V_PORTをデバイス・デスクリプタのM$Portからコピーする)，デバイス・ドライバのINITエントリをサブルーチンとして呼び出します．エラーがなければ，確保しておいたデバイス・テーブルの空きエントリに各ポインタを記録し，V$USRSを1に設定して終了します．エラーが発生した場合は，スタティック・ストレージを解放して，各モジュールをアンリンクします．

I$Attachは，ユーザが明示的に実行しなくても，I$Openなどのシステム・コールを実行すると，カーネルが内部で実行します．しかし，RAMディスクのようにデバイス・ドライバのINITで必要な初期化(RAMの確保など)を行う場合は，INIZコマンドなどでユーザが明示的にアタッチしなければなりません．そうでないと，RAMディスク上にファイルを作成しても，I$CreateでRAMディスクがアタッチされますが，ファイルがクローズされるときには，やはりカーネルが内部でI$Detachを実行するので，デバイス・ドライバのTERMルーチンが呼ばれ，せっかく確保したRAMを解放(RAMディスクが消滅)してしまうからです．

▶デバイス・テーブルのサイズ

デバイス・テーブルは，**図6**に示すように1エントリあたり18バイトの配列です．エントリの数は，起動時にカーネルによりinitモジュールのM$DevCntに書かれた数だけ確保され，initを書き換えてシステムを再起動しないかぎり変更できません．カーネルが管理する他の類似のデータ構造(プロセス・デスクリプタ・テーブル，パス・デスクリプタ・テーブル，モジュール・ディレクトリ)は，その初期サイズを超える要素を追加しようとすると，自動的にサイズが拡張されますが，デバイス・テーブルのサイズは固定で，アタッチされるデバイス数がテーブル・サイズを超えるとE$DevOvfエラーになります．

一般に，テーブル構造のデータ構造のサイズを拡張するときには，別のメモリ領域に拡張されたサイズの

〔図7〕
デバイス・テーブルの内容の例

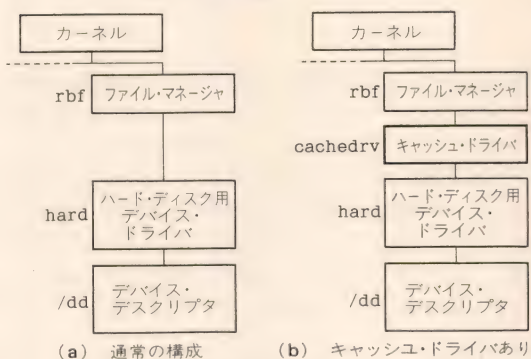
#	usrs	port	device	driver	file mgr	stat	last	busy	wake
0	5	$fffdc0	/h0	hard	rbf	$87e100	0	0	0
1	4	$fffc80	/term	VTerm	scf	$87ca50	3	3	3
2	26	$fffdc0	/dd	cachedrv	rbf	$87e100	0	0	0
3	9	$000000	/r0	ram	rbf	$87ece0	0	0	0
5	2	$fffc86	/term2	VTerm	scf	$8638b0	9	9	9
6	2	$fffc85	/term1	VTerm	scf	$864f60	4	0	0
7	2	$000000	/nil	null	scf	$87eb80	0	0	0
8	1	$000000	/pipe	null	pipeman	$87eb80	0	0	0
9	1	$fffd30	/d0	disk	rbf	$8715c0	0	0	0
10	1	$fffd30	/nd0	disk	rbf	$8715c0	0	0	0
11	1	$fffde0	/ib0	rex1120	ibf	$86ebf0	0	0	0
12	2	$fffd00	/p1	prter	scf	$86e930	7*	0	0
31	1	$000000	/cachedev	cachedrv	scf	$87b4a0	0	0	0

テーブルを確保して，もとのテーブルの内容をコピーして，テーブルのベース・アドレスを記憶しているシステム・グローバル中の変数を書き換えるという手段をとります．プロセス・デスクリプタ・テーブル，パス・デスクリプタ・テーブル，モジュール・ディレクトリなどのベース・アドレスやエントリ・アドレスは，カーネルだけが使用し，ユーザ・プログラムから参照することはできても，一般的な用途はなく，意味がないので，カーネルの都合でいつでも移動できます．

ところが，デバイス・テーブルのエントリ・アドレスは，ユーザ・プログラムがI$Attachで読み出し，後でI$Detachのパラメータとして使用できなければならず，また，デバイス・テーブルのエントリ・アドレスは，パス・デスクリプタ内(PD_DEV)にも記録され，パス・デスクリプタを使用するたびに，すなわち入出力のシステム・コールが発行されるたびに参照されるため，いつの間にか違う場所に移動していたのでは困るからです．

▶デバイス・テーブルの内容の例

図7に，実際に稼働中のシステムのデバイス・テーブルの内容を，特別のツール・プログラムで読み出した結果を示します．/ddデバイスのデバイス・ドライバがcachedrvになっていますが，これは，デバイス・

〔図8〕デバイス・テーブルを操作してキャッシュ・ドライバを組み込む

テーブルのV$DRIVを操作することにより，ファイル・マネージャ(RBF)と物理デバイス・ドライバ(hard)の間に割り込ませたディスク・キャッシュ・ドライバ(UD-CACHE)です(/h0は物理的に同一デバイスの別名だが，こちらはキャッシングしていない)．

OS-9の階層的な入出力モジュール構造は，決して固定的なものではなく，実際にはデバイス・テーブルというただ1箇所のポインタで結合されていますので，このあたりの構造をよく理解していれば，システムの稼働中でも，各モジュールを動的に再結合することができます(**図8**)．

コラム2 ファイル・マネージャを新たに開発するまでもない場合

あるファイル・デバイス群の外部(論理的)仕様の標準化と，新しいファイル・マネージャの必要性とは，必ずしも一致しないことがあります．たとえば，グラフィックスの操作は，標準化が望まれる好例ですが，標準化の程度がピクセルを単位とした点や直線，円弧，多角形などの描画のような原始的な操作では，新しいファイル・マネージャを開発するメリットはほとんどありません．なぜなら，ほとんどの操作はたんにACRTCやGDCなどの専用LSIにパラメータとコマンドを与えるだけであり，ハードウェア(LSIや画面サイズなど)に依存するため，「純粋に論理的な共通処理」はほんのわずかになってしまうからです．

このような場合は，SetStatやGetStatの機能コードとパラメータだけを決めて(pp.165-166参照)，ファイル・マネージャにはSCFなどの既存のマネージャを使用し，実際の処理はハードウェアごとに作成したデバイス・ドライバで全部行うようにしたほうが実際的です．現在，㈱フォークスから発売されている一連のOS-9マシンとアドオンCPUボードの「標準グラフィックス仕様」は，このような実現方法になっています．

ただし，デバイス・ドライバは前述のような原始的(プリミティブ)操作を実行するが，ユーザ・プログラム・インターフェースを，画面サイズや発色数のようなハードウェア・パラメータとは完全に独立にしたり，表示の拡大・縮小を可能にしたりするためには，専用のファイル・マネージャで実現するメリットがでる可能性もあります．Appendixで紹介したVivaway社のGKSMan(pp.152-153)は，この例に該当します．また，PostScriptやHPGLのような印字/プロット記述言語を解釈して，デバイス・ドライバのプリミティブ操作に分解するようなファイル・マネージャも，アプリケーション・プログラムは出力先のデバイスを気にする必要がなくなるので，開発する価値があるかもしれません．

Appendix

実際のファイル・マネージャ

最初に述べたように，OS-9では，その入出力機器の論理的な特性に応じたファイル・マネージャを作成すれば，新しいタイプのデバイスが出現しても容易に対応できます．ここでは，実際に製品として販売されている各ファイル・マネージャの特徴を，主として内部動作の観点から解説します．各ファイル・マネージャのアプリケーションから見た詳しい動作などについては，他の記事やOS-9に関する解説書，マニュアルなどを参照してください．

▶マイクロウェアから標準的に提供されるファイル・マネージャ

OS-9の開発元であるマイクロウェア社から標準的に提供されるファイル・マネージャで，組込み専用などの一部のシステムを除けば，すべてのOS-9システムに含まれています．

◈ SCF(Sequential Character File Manager)

文字単位の入出力を扱うデバイスの標準的なファイル・マネージャ．キーボードやキャラクタ・ディスプレイなどの対話環境で使用されるため，入力文字のエコー・バックや行単位の入出力処理が充実しています(**表A**)．

コンソールにかぎらず，プリンタ，マウス，モデム回線など，UnixやMS-DOSでいうところの「シリアル・デバイス」，すなわち文字単位で入出力するものなら何でもSCFが引き受けます．しかし，デバイス・ドライバとファイル・マネージャ間のデータ転送が文字(バイト)単位で行われるため，転送速度はあまり速くありません．また，通信回線で使用するためには，ブロック単位の誤り制御やプロトコル制御もサポートされていませんので，そのような処理は，アプリケーション・プログラムで行わなければなりません．

〔表A〕SCFの行編集機能

● I$Read

入力文字のエコー・バック	そのまま(PD_EKOで禁止可)
特殊文字	なし

● I$ReadLn

入力文字のエコー・バック	特殊文字と間隔以外の非図形文字を“.”に変換して表示(PD_EKOで禁止可)
特殊文字	(標準値，いずれもPD_***のオプションで変更可)

PD_BSP	1文字後退	(^H, $08)
PD_DLO	行の先頭まで後退	(^X, $18)
PD_EOR	行末	(CR, $0d)
PD_EOF	End-of-File	(Esc, $1b, 行頭でのみ有効)
PD_RPR	後退前のバッファ内容表示	(^A, $01)
PD_DUP	行バッファの内容表示	(^D, $04)
PD_BSE	BSPへのエコー・バック	(^H, $08)
PD_OVF	行バッファのあふれ警告	(^G, $07)

● I$Write

出力文字はいっさい加工なし

● I$WriteLn

行末	CRに続くLFを自動的に出力(PD_ALFで禁止可)
画面ページ	画面が一杯になると行単位一時停止(PD_PAUで禁止可)

PD_EOFに使用されるEsc文字は，SCFが主として対話的動作時にキーボードからの入力文字で強制的にEOFにするために使用するもので，CP/MやMS-DOSの^Z(CAN, $1a)のようにOS全体で定義されたものではありません．Esc以外の文字を指定することもできます．どちらかというと，Unixの^Dに近い感じです．

◈ RBF(Random Block File Manager)

フロッピ・ディスク，ハード・ディスク，RAMディスクなどのディスク・デバイスを扱います．もともとOS-9は，Unixのように高価で小容量の磁気コア・メモリと高速・大容量のディスク・パックの組合せのハードウェアを想定しているのではなく，安価で大容量の半導体メモリと低速・小容量のフロッピ・ディスク(またはディスクなし)の組合せのハードウェアに実現することを設計思想としていますので，ディスクへのプロセス・スワップなどはありません．しかしながら，OS-9においてもディスクが重要で貴重な資源であることに変わりありませんから，それを制御するRBFは，数あるOS-9のファイル・マネージャのなかでも，かなり複雑です．

RBFは，デバイス・ドライバの操作を極限まで簡素化するために，ディスク上のデータの位置をディスクの先頭から順に数えた一連のセクタ番号(LSN，論理セクタ番号)で指定し，デバイス・ドライバはLSNで指定されたセクタ単位で読み書きします．シリンダ/サーフェスといった，そのメディア(ディスク)特有のパラメータはデバイス・ドライバが処理し，ファイル・マネージャはいっさい関知しません(何も，これはRBFにかぎったことではなく，OS-9のすべてのファイル・マネージャとデバイス・ドライバはこの思想にもとづいて設計されていますが)．

RBFの管理するファイルは，階層ディレクトリ構造の中にあります．各ファイルは，必ずいずれかのディレクトリ・ファイルのエントリに属しますが，Unixのように二つ以上のディレクトリ・エントリが同一ファイルを示す「リンク」はありません．

またディレクトリ・エントリのポインタは，MS-DOSのように直接ファイルの実体を指すのではなく，「ファイル・デスクリプタ(FD)セクタ」と呼ばれるセクタを指します(Unixの*i*-nodeに相当)．FDセクタ中には，そのファイルのオーナや作成・修正日時，ファイル・サイズ，属性など

の情報と，物理的に分散している記録領域(セグメント)のサイズとディスク上の位置が書かれています．したがって，ディレクトリ・エントリ中には，ファイル名とFDセクタのLSNしか書かれていません(図A)．

RBFには多くの特徴的な操作がありますが，レコード・ロッキングはその一例です．データベースのようなアプリケーションでは，データベース・ファイルやインデックス・ファイルは，複数のプロセスから非同期に更新される可能性がありますから，何らかの相互排除処理をしないとファイルの内容が破壊されます．

たとえば，ファイル中のあるレコードを更新するためには，以下のような一連のシステム・コールを発行します．

```
os9   I$Read    レコードを読む
…               レコードの内容を変更
                (メモリ中)
os9   I$Seek    先ほどのレコード位置に戻る
os9   I$Write   更新したレコードを書き込む
```

もしあるプロセスがレコードを読み込んで，その内容を変更している間にプロセスが切り替わり，別のプロセスがそのレコード位置に書込みをしても，もとのプロセスは書込みが行われたことなどわかりませんので，そのレコードに重ね書きしてしまい，先ほどの書込みの結果が無視されてしまいます．RBFのレコード・ロッキングは，このような場合，あるレコードを先に読み込んだプロセスが，そのレコードを更新するか別のレコードを読み込むかなどを実行するまで，後からそのレコードを読み込もうとしたプロセスは自動的に待たされる(スリープ状態になる)ことになります．レコード・ロッキングは，上記のような自動処理，またはSS_LockのI$SetSttで，バイト単位でかけられます．ただし，ロックできるのはファイル中の1箇所(またはファイル全体)で，本格的なデータベースに使用するためには力不足の場合もありますが，6809の時代からサポートされていることを考えると，マイクロプロセッサ用OSとしてはかなりのものでしょう．

〔図A〕RBFのディレクトリ管理構造

ディレクトリ・ファイル
29バイト　3バイト

‥	LSN	→親ディレクトリのFDセクタ
・	LSN	→自分のFDセクタ
ファイル名	LSN	ファイル・デスクリプタ・セクタ
ファイル名	LSN	→
ファイル名	LSN	

ファイルのサイズ，属性，オーナ，作成・変更日時など(16バイト)

セグメント・リスト
[先頭LSN(3バイト) + セクタ数(2バイト)]
×48セグメント

ファイルの実体(ディスク上に分散)

- 256バイト/セクタ
- ルート・ディレクトリでは“‥”と“・”は同じ
- LSNは3バイトで表される
- ファイル名の最後のバイトはMSBがセットされている(6809版との互換性のため)
- 削除されたディレクトリ・エントリの先頭バイトは$00

◆ PipeMan(Pipe File Manager)

メモリ中に確保されたFIFO(First In First Out)バッファを管理します．通常は，書込みと読出しを異なるプロセスが並行して行うため，キメの細かい競合(排他)処理を行います．このFIFOバッファは，パイプの作成(create)時にサイズが指定されていれば，そのサイズのバッファがシステム・メモリから確保されますが，サイズが指定されていない場合は，デフォルトとしてパス・デスクリプタ中の90バイトの領域がバッファとして使用されます．

パイプは，通常SCFやRBFのデバイスが接続される標準入出力パスにリダイレクトされることが多いので，アプリケーション・プログラムからこれらのデバイスと見かけが同じになるようなシステム・コールがサポートされています．とくに，名前つきパイプでは，RBFファイルと同じ形式の疑似的なファイル・デスクリプタ・セクタを読み込んだり，ファイルのアトリビュートを設定でき，DIRやATTRなどのユーティリティ・コマンド・プログラムが使用できるようになっています(表B)．

パイプでは，すべての操作はファイル・マネージャで実行され，ファイル・マネージャがデバイス・ドライバを呼び出すことはありません．しかしながら，実際には何もしないダミーのデバイス・ドライバ(nullドライバ)が必要です．その理由は二つあります．一つは形式を整えるためです．カーネルは，それがパイプだからといって特別扱いしません(というよりも，カーネルはパイプが何であるかということに関知しない)ので，他のファイル・マネージャの場合と同様のデータ構造(デバイス・テーブル・エントリ)を管理するためにデバイス・ドライバが必要です．もう一つ

〔表B〕パイプでサポートされるGetStatとSetStat
(カッコ内は対応するファイル・マネージャ)

● I$GetStt		
SS_Ready	バッファ中のデータ・バイト数を検査	(SCF)
SS_Size	バッファ・サイズを検査	(RBF)
SS_EOF	EOFを検査	(RBF)
SS_FD	疑似ファイル・デスクリプタ・セクタの読込み	(RBF)
● I$SetStt		
SS_Size	0ならばバッファをリセット	(RBF)
SS_FD	何もしない	(RBF)
SS_Attr	パイプのアトリビュート(読み書きの許可)を変更	(RBF)
SS_SSig	データ・レディ時に発生するシグナルの登録	(SCF)
SS_Relea	SS_SSigで登録したシグナルの取り消し	(SCF)

は，デバイス・ドライバのモジュール・ヘッダのM$Memフィールドにパイプ・デバイスが使用するデバイス・スタティック・ストレージのサイズが書かれているからです．

パイプに書き込むプロセスが一つもなく，バッファが空になるとそのパイプを読み出すプロセスにはEOFが返されます．

▶マイクロウェアからオプションで提供されるファイル・マネージャ

以下のファイル・マネージャは，OS-9の標準システム(ポート・パックなど)には含まれておらず，必要に応じて別契約で購入します．

◈ NFM(Network File Manager)

サーバ/クライアントの区別のない，完全に平等なネットワークをサポートします．サーバとクライアントの区別がないということは，逆にいえば，すべてのステーションが同時にサーバやクライアントになるということです．ネットワークの向う側にあるリモート・ステーションに属するファイル・デバイスをアクセスするためには，そのステーション上で走行するダミー・プロセスにアクセス要求を出します(図B)．このダミー・プロセスはファイル・マネージャ(NFM)が生成し，ユーザ・プログラムではとくに気にする必要はありません．ダミー・プロセスからさらにリモート・アクセスをすることはできません．

◈ SBF(Sequential Block File Manager)

MTなどのテープ・デバイスをサポートするファイル・マネージャです．ある程度の量のデータを連続的に転送する必要があるテープ・デバイスの特性に適合させるため，アンバッファド・モードとバッファド・モードがあります(図C)．アンバッファド・モードでは，入出力を要求したプロセスは実際のデータ転送が完了するまでブロックされます(通常の入出力動作と同じ)．バッファド・モードでは，プロセスはファイル・マネージャが確保したバッファ・メモリとの間でデータをコピーするだけで，実際のデータ転送はファイル・マネージャが起動したダミー・プロセスが行います．

◈ SBBF(Sequential Binary Block File Manager)

マイクロウェア・ジャパン㈱の開発による，A-D/D-Aなど従来のファイル・マネージャで扱いにくかったデバイスの操作用．SCFとは違い，ファイル・マネージャとデバイス・ドライバ間のデータ転送をブロック単位で行うので高速ですが，行編集などのデータの加工は，いっさい行いません．ファイル・マネージャとして最低限の機能を備え，デバイス・ドライバさえ用意すればどんなデバイスでも操作できますが，逆にすべての処理はデバイス・ドライバが行わなければなりません．

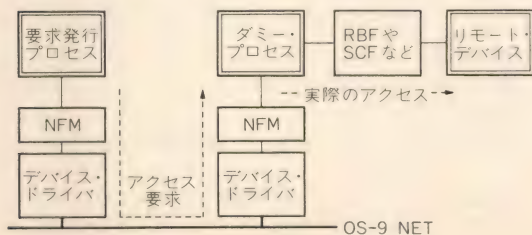
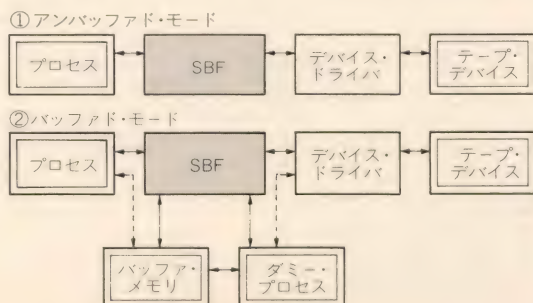
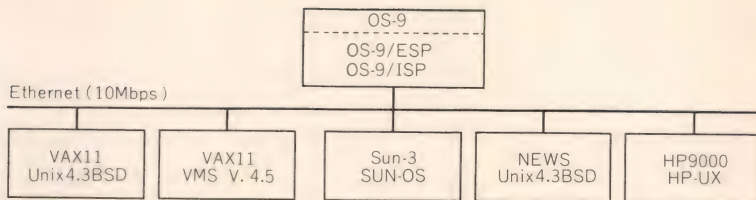
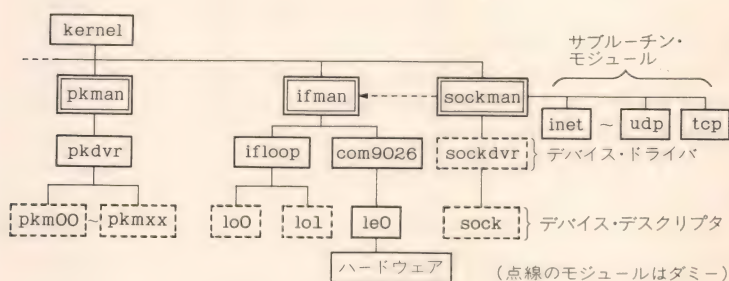
◈ SockMan(Socket File Manager)

マイクロウェアから発売されているOS-9/ISP(Internet Support Package)に含まれ，Unix BSD 4.3相当のソケットによるネットワーキングをサポートします．

SockManは，ほぼトランスポート層とネットワーク層をサポートしているようです．ただし，SockManは(狭い意味での)特定のプロトコルをサポートしているわけではなく，TCP(トランスポート層)やIP(ネットワーク層)のプロトコルは，それぞれファイル・マネージャから呼び出される独立したサブルーチン・モジュールで実現されています．このため，X.25などの別のプロトコルを容易にサポートできるようになっています．

SockManとは別に，IFMan(Interface File Manager)という別のファイル・マネージャがあり，ハードウェアに依存しないネットワークの診断(ICMP：Internet Control Message Protocol)とシステム・コンフィギュレーション(アドレスの変更・削除・追加)を行うことになっています(図E)．また，実際にハードウェアを制御するデバイス・ドライバは，IFManの下に位置していて，SockManのデバイス・ドライバ(SockDrv)は中味のないダミー・ドライバですから，SockManは何らかの形(たとえばパスをオープンするなど)でIFManを呼び出しているはずですが，詳細は不明です．このIFManのデバイス・ドライバは実際にハードウェアを操作するわけですが，Internetアドレスと物理ノード・アドレスの変換(ARP＝Address Resolution Protocol)もデバイス・ドライバが行っていますから，デバイス・ドライバの変更だけで，Ethernetはもとより，ArcNet, SDLC, point-to-pointのデータ通信などの多くのネットワークに対応できます(図D)．

OS-9/ISP(ESP)には，この他にもPKMan(Pseudo

〔図B〕NFMを経由したリモート・デバイス・アクセス

〔図C〕SBFのアンバッファド・モードとバッファド・モード

〔図D〕
OS-9/ISP(ESP)によるネットワーク

〔図E〕
OS-9/ISP の構造

Keyboard File Manager)というファイル・マネージャが含まれています. PKMan は, telnet で Unix 側からログインされたときに Ethernet を経由して疑似的なキーボード・インターフェースを提供するファイル・マネージャで, Unix 側の ptty に相当します.

アプリケーション・レベルでは, FTP(File Transfer Protocol)と telnet がサポートされており, Unix システムと OS-9 システム間で相互にファイル転送とログインができます. NFS はまだサポートされていないようです.

この OS-9/ISP は, 以前から発売されていた OS-9/ESP (Ethernet Support Package)のスーパセットです(文献7および本誌別稿参照). OS-9/ESP は, CMC(Computer Machinery Company)社製の ENP-10 というハードウェア専用でしたが, 主要ファイル・マネージャの ENPMan が改良され, 特定のハードウェアに依存しない SockMan になりました. 現在マイクロウェアから出荷されているのは, モトローラ社の MVME147 に実装された LANCE チップ・セット用のデバイス・ドライバです.

このようなネットワーキングを OS-9 で実現しようとすると, カーネル/ファイル・マネージャ/デバイス・ドライバ/デバイス・デスクリプタの4段階のモジュール構造では, 不足の感じがします. サブルーチン・モジュールを導入したのは, 苦肉の策でしょうか. もっとも, 他の OS では, そもそもこのような物理的に分離できるモジュール構造は存在しませんから, それらに比べるとはるかに優れているのですが.

▶専用ファイル・マネージャ

以下のファイル・マネージャは, マイクロウェアにより開発されたものですが, 現在のところ特定のシステム専用で, 一般的な販売は行われていないようです.

◈ **UCM**(User Communications Manager)

フィリップス社との共同開発による CD-I(Compact Disk Interactive)システムに使用されている CD-RTOS (Compact Disk Real Time Operating System)で, キーボードやマウス, ビデオ, オーディオなどのいっさいのユーザ・インターフェースをつかさどります.

◈ **CDFM**(Compact Disk File Manager)

Mode2(Green Book)の CD 上のファイル管理を行います. CD-I ディスクは読出し専用なので, ファイルの作成・削除, レコード・ロッキングなどの問題はありませんが, CD が磁気ディスクに比べてランダム・アクセスが苦手(低速)なので, 以下のような手法を採用しています.

① パス・テーブル

ディスク上のすべてのファイル名とその開始位置を記録したファイルで, ディスクのマウント時にメモリに読み込まれる. 階層ディレクトリの深い部分のファイルをアクセスするときに, 多くのディレクトリをたぐる必要がなくなる.

② エレベータ・ロジック(図F)

あるプロセスからの要求でディスク上をシーク中に, 別プロセスからシーク(SS_Seek の Setstat)要求があった場合, 先に到達する位置の処理を優先する(他のファイル・マネージャでは, 先に要求したプロセスの処理が終了するまで後から要求したプロセスの実行は保留される). デパートなどのエレベータと同じ運行方法(文献4参照).

◈ **NRF**(Non-volatile RAM Disk File Manager)

これも CD-I 用のファイル・マネージャで, ゲームの得点などを不揮発性メモリにファイルとして記憶します. RBF の管理化にある RAM ディスクに似ていますが, 高価で少量(8Kバイト程度)のメモリを使用するために, 経済的な構造をしています. まだ実際にリリースされていないので

内容は不明ですが，以下のような処理を行っていると想像できます．

① 一つのファイルはつねにメモリ上で連続した領域にある(マップ情報が不要)．

② ファイルが削除されるとリパックされる(記憶領域が断片化しない)．

パイプの場合と同様に，すべての操作はファイル・マネージャで行われ，デバイス・ドライバはダミーです．CD-Iの仕様書(Green Book)によれば「パイプより低速」です．

◈ **XF**(X68000 File Manager)

UCMをベースに開発された，シャープX68000のOS-9のユーザ・インターフェースの制御専用ファイル・マネージャ．オーバラップ型マルチウィンドウやマウスをサポートします．実際には，ほとんどの主要な処理はデバイス・ドライバで行っています．

▶マイクロウェア以外の開発したファイル・マネージャ

以下のファイル・マネージャは，マイクロウェア以外の会社で開発され，販売されています(注1)．

◈ **IBF**(IEEE488/GPIB File Manager)

筆者が開発したIEEE488/GPIBインターフェース・バス

〔図F〕CDFMのエレベータ・ロジック

〔図G〕IBFのシグナルによるプロセスの同期

専用のファイル・マネージャ．以下のような特徴をもちます．

- IEEE488.1-1987に定義されているほとんどの機能をサポート．
- DMAによる高速ブロック転送可能．
- SRQやデバイス・クリアなどの事象でシグナルを発生し，プロセスの同期をとれる(**図G**)．
- デバイス・ディペンデント・メッセージ(データ)の転送は，RBFやSCFのデバイスと同様にread()/write()/printf()/gets()などの標準Cライブラリ関数で行える．
- IEEE488/GPIBに特有の機能は，ヒューレット・パッカード(HP)社のHP-UX(Unix)用DIOライブラリと互換のCライブラリ関数でサポート(**図H**)．
- ListやEchoなどのOS-9の標準コマンドや標準入出力のリダイレクトなど，シェル・レベルでの使用が可能(プログラム作成前のテストなどに使用)．

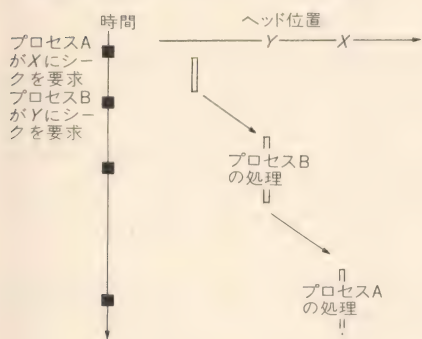
〈例〉 $ echo hello >/ib0/11; listln /ib0/11

現在IBFは，特定のハードウェア用にインプリメントされたオブジェクト・コード・ライセンスと，自社のハードウェアにインプリメントするためのソース・コードを含むOEMライセンスの2種類の形態で供給されています．

◈ **GKSMan**(Graphics File Manager)

国際的なグラフィックスの規格であるGKS(Graphics Kernel System)をサポートします．ISO 7942-1985のレベル2cのサブセットの機能(整数のピクセル座標のみ)をもちます．デバイス・ドライバは，**表C**に示すように，通常の7個のエントリ(注2)から拡張された合計39個のエントリをもちます．デバイス・ドライバが実際にはそのエントリ

(注1) いずれも㈱エー・アール・ケー・コーポレーション(0484-45-9020)から入手可能

(注2) D$TRAPは，マニュアルでは「将来のバージョンでは必要になるかもしれないのでオフセット値を0にしておくように」と書かれているが，正式にはまだサポートされていない．

〔図H〕IBFのC言語用ライブラリ

_ib_abort()	IFC送出
_ib_clear()	デバイス・クリア送出
_ib_ppoll()	パラレル・ポール
_ib_spoll()	シリアル・ポール
_ib_bus_status()	バス状態チェック
_ib_lock()	インターフェースのロック
_ib_rqst_srvce()	SRQ発生
_ib_status_wait()	事象発生待ち

をサポートしていない(エントリのオフセット値が0または「実行できない」エラーを返す)場合は，ファイル・マネージャ(GKSMan)が他のエントリを使用してシミュレートします．当然のことながら，シミュレーションでは速度が低下しますが，デバイス・ドライバが簡単になります．デバイス・ドライバがもつべき最小限の機能は，直線の描画と描画モードをEx-ORにできることだけです．C言語から簡単に実行できるライブラリが付属します．英国Vivaway社(注3)の開発．

◆ **VBF**(Variable Block File Manager)

従来SCFが使用されてきたシリアル回線による通信を効率化するために，パケット単位で通信するように設計されています．SCFでは，あるプロセスがI$Read/I$ReadLnシステム・コールを発行してデバイス・ドライバ内でデータの到着を待っているときに，他のプロセスがそのデバイスからデータを送信しようとすると，SCFの相互排除処理のため受信プロセスがデータを受信するまで待たされますが，VBFでは受信と送信の両方のプロセスが同時にデバイス・ドライバに入ることを許しています．通信相手からの呼出しや応答をモニタする受信プロセスと，相手に対して呼出しや応答をする送信プロセスを並行動作させるような通信目的に使用できます．英国Vivaway社の開発．

◆ **MCF**(Multi-channel Character File Manager)

VBFにSCFと同等の行編集機能を追加したもので，ターミナルでの使用も可能．

◆ **MFM**(Memory File Manager)

RBFによるRAMディスクに似ていますが，RAMディスクではあらかじめディスク・サイズぶんのメモリを確保するのに対して，必要に応じて動的にシステム・メモリを要求・返却するのでメモリ使用効率が非常によいのが特徴です．マルチレベル・ディレクトリやレコード・ロッキングをサポートしないなど，名前付きパイプに似ていますが，FIFOではないので，データを読み出しても消えず，書込み・読出しプロセスがブロックされることもありません．RBFに比べて高速で，コンパイル/アセンブルの一時ファイル作成やデータベースのインデックス・ファイルなどに最適です．また，ROM化されたシステムでも応用できるように，特殊形式のデータ・モジュールを読出し専用ファイルとみなすこともできます．米国Windsor社の開発．

〔表C〕GKSManの拡張ドライバ・エントリ
(＊は現在サポートされていない)

オフセット	ニモニック	動作	
$00	D$INT	デバイスを初期化する	OS-9の共通エントリ
$02	D$READ	デバイスから読み込む*	
$04	D$WRITE	デバイスに書き込む*	
$06	D$GSTA	デバイスの状態を得る	
$08	D$PSTA	デバイスの状態を設定	
$0a	D$TERM	デバイスを終了する	
$0c	D$TRAP	バス・エラーまたはアドレス・エラー	
$0e	D$OPENPATH	デバイスへのパスをオープン	GKSMan固有のエントリ(拡張部分)
$10	D$CLOSPATH	デバイスへのパスをクローズ	
$12	D$SETPLOT	プロット・モードを設定	
$14	D$MARK	マーカを描く	
$16	D$LINE	直線を描く	
$18	D$CHAR	文字を描く*	
$1a	D$FRECT	長方形を描く(塗りつぶし)	
$1c	D$CLEAR	画面全体を消去	
$1e	D$PMARK	多重マーカを描く	
$20	D$PLINE	多角線を描く	
$22	D$FILL	範囲を塗りつぶす	
$24	D$TEXT	テキストを描く	
$26	D$CIRCLE	円/楕円を描く*	
$28	D$ARC	孤を描く*	
$2a	D$DONE	動作の終了	
$2c	D$PIXLIN	ピクセルによる直線	
$2e	D$HIGH	ハイライトのオン/オフ	
$30	D$POINTER	ポインタ位置と形式	
$32	D$PICK	ピック・モードのオン/オフ	
$34	D$POPWIND	ポップアップ・ウィンドウ	
$36	D$BITBLT	BitBlt動作	
$38	D$CURMOVE	カーソルをX/Yに移動*	
$3a	D$LINEID	行の挿入/削除*	
$3c	D$CHARID	文字の挿入/削除*	
$3e	D$EOLCLR	行の終わりまで消去	
$40	D$EOSCLR	画面の終わりまで消去*	
$42	D$CLR	画面を消去*	
$44	D$HOME	カーソルをホーム・ポジションに移動	
$46	D$CONTROL	制御文字*	
$48	D$ATTR	アトリビュートの制御	
$4a	D$SCROLL	スクロール*	
$4c	D$FONT	フォントの組込み/取外し	

(注3) 現在Vivaway社は業務を停止し，Snowtop Computer社が後を継いでいる．

II　ファイル・マネージャの構造と設計

3 ファイル・マネージャの構造

ファイル・マネージャは，デバイス・ドライバとは違い，ユーザ・サイドで作成する必要がほとんどないので，その構造についての情報はあまり多いとはいえません．しかしマイクロウェアから供給されるマニュアルや各種定義ファイルとデバッガを駆使すれば，その構造を理解することは，それほど難しくありません．

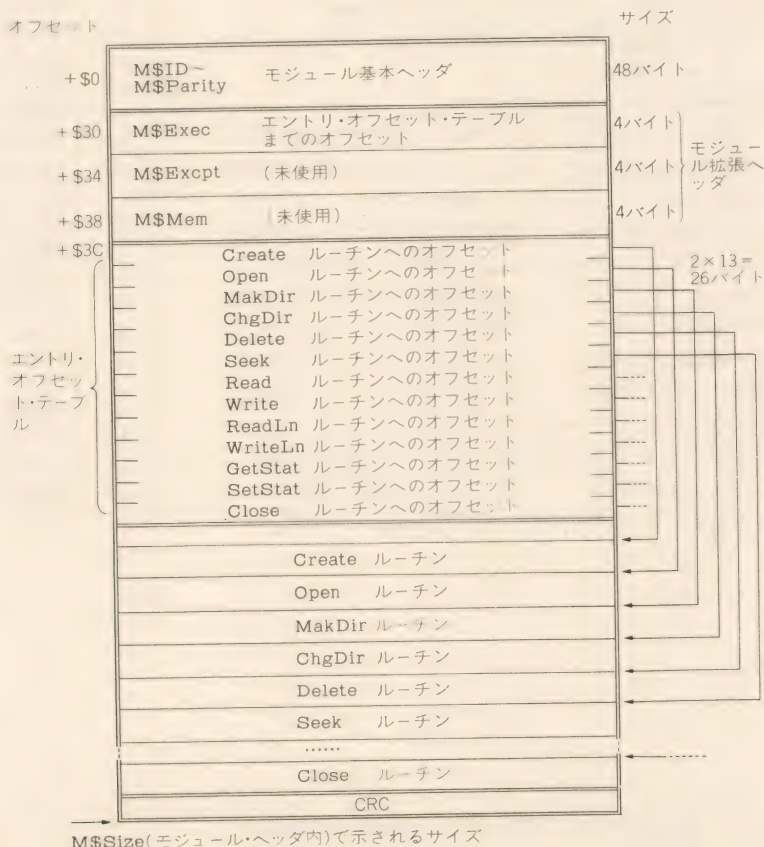
以下の説明中でE$***はエラーの理由を表すエラー・コードです．F$***はOS-9の入出力以外のシステム・コール，I$***は入出力関係のシステム・コールです．R$**はユーザ・レジスタ・スタック(システム・コールを発行したときの実レジスタのメモリ中のコピー)中のレジスタを表します．これらのニモニック(ラベル)は，ライブラリ・ファイル sys.l の中で定義されています(pp.144-145 コラム1参照)．

3.1　カーネルとのインターフェース

あらゆるファイル・マネージャは，少なくともカーネルとのインターフェースから見るかぎり，すべて平等です．だからこそ，そのファイル・デバイスの論理的・物理的構造がどのようになっているかに関係なく，ユーザ・プログラムからあらゆるファイル・デバイスを統一的に扱うことのできる「ユニファイドI/O」が実現できるともいえます．

▶エントリ

すべてのファイル・マネージャは，図9のようにCreate～Closeの13個のエントリをもちます．各エ

〔図9〕
ファイル・マネージャの一般的な構造
(各ルーチンの実際の位置は図のとおりでなくてもよい)

ントリは，エントリ・オフセット・テーブル内に記述されたエントリ・オフセット・テーブルの先頭からのオフセット(ワード，2バイト)により，その位置を知ることができます．エントリ・オフセット・テーブル自身のファイル・マネージャ・モジュールの先頭からのオフセットは，モジュール・ヘッダ内のM$Exec(ロング・ワード，4バイト)に記述されています．

ファイル・マネージャの各エントリは，カーネルがI$***システム・コールを実行するときに必要に応じてサブルーチンとして呼び出され，ユーザ・プログラムが直接ファイル・マネージャを呼び出すことはありません(呼び出すことはできない)．

▶パラメータ

ファイル・マネージャの各エントリがカーネルから呼び出されたときには，以下のレジスタが標準的に設定されています．これらのレジスタは，カーネル側でスタック中に保存されていますから，ファイル・マネージャ内で破壊してもかまいません．

(a1)　パス・デスクリプタのアドレス
(a4)　プロセス・デスクリプタのアドレス
(a5)　ユーザ・レジスタ・スタックのアドレス
(a6)　システム・グローバル領域のベース・アドレス

MakDir，ChgDir，Deleteの各エントリは，呼出しプログラムとパス番号のやりとりをしないので，明示的なパスが存在しないように見えます．しかし，実際には，後述のこれらのエントリの動作の解説を参照すればわかるように，ファイル・マネージャではパスを操作しなければなりませんので，カーネルは一時的なパス・デスクリプタを割り当てます．

各エントリに対応するシステム・コール(I$***)をユーザ・プログラムが発行したときに与えたパラメータ(パス・リストのアドレスやバッファ・サイズ，SS_***機能コードなど)は，すべてa5レジスタで指されるユーザ・レジスタ・スタック中にあります．参照するときは以下のようにします．

```
move.b  R$d0+3(a5),d0   入力パラメータのd0.bを得る
```

ユーザ・レジスタに値を返すときは，メモリ中のユーザ・レジスタに書き込まなければなりません．MPUの実レジスタに書き込んでもダメです(エラー発生時のエラー・コードとキャリ・ビットを除く)．

ファイル・マネージャからカーネルに戻るときには，二つの場合があります．一つは指示された操作が問題なく実行できた場合であり，必要ならばユーザ・レジスタ・スタック中に戻りパラメータを書き込み，条件コード・レジスタのキャリ・ビットをクリアしてrts命令を実行します．

●エラーなしのリターン

```
moveq.l  #0,d1        キャリをクリアする
rts                   カーネルに戻る
```

入力パラメータが論理的に矛盾しているなど，何らかの理由で指示された操作が失敗した場合は，エラーの理由を示すエラー・コード(E$***)をd1.wレジスタに入れ，条件コード・レジスタのキャリ・ビットをセットして，やはりrts命令でカーネルに戻ります．いずれの場合も，ユーザ・レジスタ・スタックに値を返す必要はありません．

●エラーありのリターン

```
ErrFull: move.w  #E$Full,d1  エラー・コード
                             をセット
         ori.w   #Carry,sr   キャリをセット
         rts                 カーネルに戻る
```

ファイル・マネージャから呼び出され，物理的な操作を実行したデバイス・ドライバがエラーを返した(キャリがセットされている)ときには，すでにデバイス・ドライバがd1.wレジスタにエラー・コードを入れていますから，キャリをセットしたままカーネルに戻ります．

ReadやWriteのエラーではカーネルはとくに何もしませんが，CreateやOpenのエントリでエラーが発生すると，カーネルはファイル・マネージャを実行する前に確保したパス・デスクリプタを解放し，パスを閉じます．パス・デスクリプタやスタティック・ストレージのメモリの確保や解放はカーネルが行います．ファイル・マネージャは，独自のバッファなど以外のメモリ領域の確保や解放を行う必要はありません．

3.2　各エントリの動作

以下に，ファイル・マネージャの13個のエントリの動作について解説します．ファイル・マネージャによっては，すべてのエントリをサポートしていない場合もあります．サポートされていないエントリはE$UnkSvc(該当機能なし)のエラーを返します(Seekを除く)．以下でエントリ名の後のかっこ内のI$***は，対応する入出力システム・コールです．

▶ Create(I$Create)

このエントリは，新しいファイルを作成し，オープンします．ディスクなどのように，一つのファイル・デバイス上に複数の名前付きファイルが存在できる「マルチファイル・デバイス」に対してのみ意味があります．マルチファイル・デバイスとは，別の見方をすれば，ディレクトリ構造をもつデバイスのことです．そうでないデバイス(SCF など)では，Open のエントリと同じにします．マルチファイル・デバイスでも，CD のように読出し専用の場合は，Create のエントリはサポートされません(E$UnkSvc または E$BMode エラーを返す)．

◇処　理

まずパス・リスト(R$a0)を解析して，文法的に正しく，ファイルを作成するディレクトリが存在して，書込み許可があることを検査します．メディアの空き領域が十分あり，ファイルが作成できるならば，アトリビュート(R$d1.w)や初期サイズ(R$d2.l)，オーナ(P$GID，P$UID)などのパラメータで初期化します．0 以外の初期サイズを指定したときのファイルの内容は，一般的にはファイル・マネージャの自由ですが，不定のままにしておいたほうがよいでしょう．そうでないと，たとえば RBF ではファイルの初期内容は不定ですから，ファイルの初期内容に依存したアプリケーション・プログラムは RBF に対しては動作しなくなるおそれがあります．Create のエントリは，パス・リストの解析など，多くの部分が Open の処理と共通です．

◇既存ファイルの Create

OS-9 の I$Create による新ファイルの作成では，Unix の creat( ) システム・コールとは違い，マルチファイル・デバイスですでに存在するファイルを「作成」しようとすると，エラー(E$CEF)になります(Unix の creat( ) ではファイルの長さを 0 に切り捨てるが，エラーにはならない)．OS-9 の C コンパイラに付属の creat( ) 関数は，直接 I$Create システム・コールを発行せず，シミュレーションを行っています．I$Create に直接対応するのは，create( ) という別のライブラリ関数です．

◇ディレクトリ・ファイルの作成

Create のエントリは，RBF のように階層化ディレクトリ構造をもつファイル・デバイスに新しいディレクトリを作成するためには使用できません．ディレクトリの作成は MakDir で行います．したがって，入力パラメータのアトリビュート・バイト(R$d1.w)の Dir ビットがセットされているときは，E$BMode エラーを返します．

◇初期サイズ

アクセス・モード・バイト(R$d0.b)の I ビット(ビット 5)がセットされているときは，R$d2.l で作成するファイルの初期サイズが与えられていますので，そのぶんの記憶領域を確保します．ただし，この機能をサポートしているのは，RBF や Pipe など，実際に記憶の目的で使われるファイル・デバイスだけです．SCF などでは，I ビットがセットされていても無視します．

具体的には，RBF ではビット・マップを検索して指定されたサイズの連続した空き領域を確保します．ただしこの空き領域の検索は，複数のビット・マップ・セクタにまたがることはありません(1 ビット・マップ・セクタで表せるサイズを超えるぶんは切り捨てられる)．

Pipe では，デフォルトの 90 バイトのバッファ(パス・デスクリプタの一部)の代わりに，指定されたサイズのメモリを F$SRqMem でシステムから確保します．COPY コマンドは，コピー先のデバイスにファイルを作成するときに，ソース・ファイルのサイズを初期サイズとして指定しますので，以下のようなコマンド・ラインはすぐに実行が終わります．

```
$ copy  /dd/startup  /pipe/copied
```

しかし，Shell の＞によるリダイレクションは初期サイズを指定しませんから，以下のようなコマンド・ラインは Shell に制御が戻ってきません．

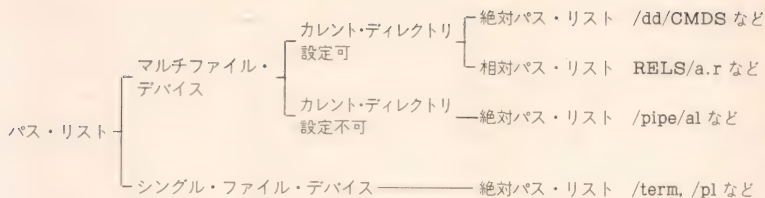
```
$ list  /dd/startup  >/pipe/listed
```

(いずれの場合も/dd/startup は十分大きいものとする)

上記のようにブロックされた LIST コマンドを ^E で強制終了させるか，^C でバックグラウンド・プロセスにして，dir -e /pipe を実行してみると，デフォルトのパイプ・サイズが 90 バイトであることがわかります．

▶ Open(I$Open)

すべてのファイル・デバイスは，そのデバイスに対するパスをオープンしないと使用できませんから，この Open エントリは，すべてのファイル・マネージャがサポートします．マルチファイル・デバイスでは，Open のエントリでオープンできるのは，すでに存在

〔図10〕
パス・リストの種類

するファイルだけです.

◇パス・リスト(pathlist)の解析

Create と同様に, Open のエントリの主要な仕事の一つは, パス・リストの解析です. パス・リストの構成は, マルチファイル・デバイスとシングル・ファイル・デバイスで異なります. マルチファイル・デバイスでは, カレント・ディレクトリを設定できる場合(ChgDir をサポートしている場合)は, / から始まる絶対パス・リストと, カレント・ディレクトリを基点とした相対パス・リストの二つの場合があります. シングル・ファイル・デバイスやカレント・ディレクトリを設定できないマルチファイル・デバイスの場合は, パス・リストはつねに / から始まる絶対パス・リストです(**図10**).

パス・リストの解析には F$PrsNam システム・コールを使用します. パス・リストは, Unix と同じように, / で区切られた「要素」(エレメント)から構成されます. F$PrsNam システム・コールは, 現在のポインタの位置から一つぶんの正当なパス・リストの要素を切り出しますから, マルチファイル・デバイスのようにパス・リストが複数の要素から構成されている場合は, F$PrsNam を何回も実行します.

(1) 絶対パス・リスト

シングル・ファイル・デバイスの場合は, デバイス名の後にはもうパス・リストの要素はないはずですから, 先頭の要素が正当なデリミタ文字(スペース, タブ, キャリッジ・リターン, ライン・フィード, ナルなど)で終了していることを検査します. もし不正なデリミタの場合は, E$BPNam エラーになります.

マルチファイル・デバイスの絶対パス・リストは, モジュール名の後に(/ で区切られて)ファイル名がつづきます. 階層ディレクトリ構造をサポートしているディレクトリの場合は, さらにディレクトリ名とファイル名がつづきますから, デリミタ文字が現れるまで解析をつづけると同時に, 各要素の正当性(ディレクトリやファイルが存在しアクセス許可があること)を検査します. 絶対パス・リストの場合は, つねにルート・ディレクトリから検索します. 途中や最後のディレクトリやファイルが存在しない場合は, E$PNNF エラー(パス・ネームが見つからない)を返し, 存在していてもオープンしようとしているプロセスがアクセス許可をもたない場合は, E$FNA(ファイルをアクセスできない)エラーを返します.

(2) 相対パス・リスト

相対パス・リストは, ChgDir 可能なマルチファイル・デバイスにのみ許されます. パス・リストの解析の基点が必要ですから, プロセス・デスクリプタの P$DIO から必要な情報を得ます. P$DIO には, カレント・データ・ディレクトリとカレント実行ディレクトリの両方のフィールドがありますが, どちらを基点とするかは, 入力パラメータのアクセス・モード(R$d0.b)実行ビット(ビット 2)がセットされているかどうかで選択します. 基点以降のパス・リストの解析は, 絶対パス・リストの場合と同じです.

(3) パス・リストの更新

Open にかぎらず, ファイル・マネージャでパス・リストを解析したら, 「パス・リストの更新」を行います. これは, OS-9 のプログラムの慣習として, キャリッジ・リターンまたはナル文字で終結するまでの行に, スペースで区切られた複数のパス・リストが列挙された場合に, 処理プログラムの操作を簡単にするためです(ただし, この処理で便益を得るのはアセンブリ言語で書かれたプログラムだけで, C で書かれたユーザ・プログラムではほとんど関係がない). ファイル・マネージャの各エントリのうち, パス・リストを入力パラメータにもつ Create, Open, MakDir, ChgDir, Delete の各エントリは, カーネルに戻るときに, デリミタの先にまだ有効なパス・リストがあるならば, ユーザ・レジスタのパス・リストのポインタをつぎのパス・リストまで進めます(**図11**).

(4) エンタイア・オープン

RBF の場合, デバイス名の直後またはパス・リスト

の先頭に @ があるときには，ディスク全体を一つのファイルと見立てて「エンタイア・オープン」します．この方法は，FREE コマンド(ディスクの空き領域をみるコマンド．全体のセクタ数および未使用のセクタ数を表示する)がディスクのビット・マップの状態を読み込んだり，事故で壊れたディレクトリ構造を修復したりするために使用されます．

カーネルがパス・リストから必要な情報を得るメカニズムについては，コラム3を参照してください．

◇アクセス・モード

I$Open や I$Create のエントリでパスをオープンするときには，R$d0.b でアクセス・モードを与えます．I$ChgDir や I$MakDir，I$Delete などはユーザにパスは返りませんが，カーネルとファイル・マネージャでの処理はオープンの処理とほとんど変わりませんので，アクセス・モードも同じように扱われます．アクセス・モードの各ビットは，以下のように定義されています(複数ビットの組合せ可)．

ビット 0 R　読出しモード
　　1 W　書込みモード
　　2 E　実行モード
　　3 —　(未定義)
　　4 —　(未定義)
　　5 I　初期ファイル・サイズ
　　　　(I$Create と I$MakDir のみ)
　　6 S　非共有オープン
　　　　(I$Create と I$Open のみ)
　　7 D　ディレクトリ・モード(I$Open のみ)

これらの各アクセス・モード・ビットのうち，S(非共有オープン)ビットを除く，R，W，E，D の各ビットは，カーネルにより，オープンするデバイスのデバイス・デスクリプタの M$Mode バイトの対応するビットがセットされている(そのアクセス・モードが許可されている)かどうかのチェックを受けます．もし許可されていないアクセス・モードが与えられた場合は，E$BMode エラーでカーネルから直接呼出しプログラムに戻ります(ファイル・マネージャは実行されない)．また，もしパス・リストが相対パス・リストの場合は，このチェックに先立ち，アクセス・モードの E ビットがセットされているかどうかで，どちらのデフォルト・デバイスを使用するのかを決めます．R$d0.b は，パス・デスクリプタの PD_MOD にコピーされ，ファイル・マネージャに渡されます．

〔図11〕パス・リストの更新

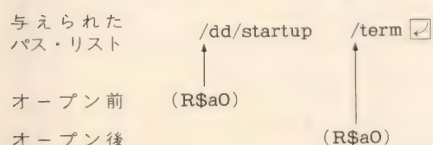
アクセス・モードについてカーネルがチェックするのは，M$Mode で許可されているかどうかだけですから，後の処理はファイル・マネージャによります．たとえば，I(初期ファイル・サイズ)ビットは，RBF と Pipe 以外のほとんどのファイル・マネージャでは無視されます．R と W ビットは，RBF と Pipe で実際のファイルのアトリビュートとオーナ ID，および I$Open を発行したプロセスのオーナ(P$UID と P$GID)を比較して，アクセス許可があるかどうかのチェックに使用されます．S(非共有オープン)ビットは，カーネルでチェックされずにそのままファイル・マネージャに渡されます．S ビットについては，つぎの項で説明します．

アクセス・モードの W(書込み)ビットがセットされていて，ファイル・デバイスにタイム・スタンプ機能があるときには，ファイル・マネージャが最終変更日時を更新します．

◇オープン時の排他制御

OS-9 はマルチプロセスの OS ですから，同一ファイルを複数のプロセスがオープンすることがあります．ところが，書込みのできるファイル・デバイスではデータの内容の正当性を保障するために，また読出しのできるファイル・デバイスではデータを他のプロセスに横取りされないことを保障するために，オープンできるプロセスの数を制限したい場合があります．OS-9 では，ファイルのオープン(I$Create と I$Open)時に以下のような排他制御を行っています．

(1) カーネルの行う排他処理(M$Mode の S ビット)

オープンするデバイスのデバイス・デスクリプタの M$Mode の S ビットがセットされている場合(非共有デバイス)は，カーネルはそのデバイスに対してオープンできるパスをただ一つに制限します．そのデバイスにすでにパスがオープンしていて，二重にオープンしようとすると E$DevBsy エラーになります．後からオープンしようとしたプロセスは，エラーを受け取りそのデバイスの使用をあきらめるか，または

F$Sleepで一定時間経ってから再試行します．

カーネルは，すでに別のパスがオープンしているかどうかを，非共有デバイスのデバイス・スタティック・ストレージのV_Paths(パス・デスクリプタのリンク・リスト)を調べ，判断します．

この方法はデバイス単位で行われるため，SCFのようなシングル・ファイル・デバイスで，プリンタやプロッタの出力に複数プロセスの出力が混じらないようにしたり，シーケンシャルな入力を別のプロセスに横取りされないようにするために使用されます．

(2) ファイル・マネージャの行う排他処理(アクセス・モードとファイル・アトリビュートのSビット)

この方法は，デバイス全体を共有不可にしてしまうのは具合の悪い場合に使用します．ということは，容易に想像できるとおり，マルチファイル・デバイス専用です．RBFでは，アクセス・モード(R$d1.w)またはファイル・アトリビュート(FDセクタのFD_Attr)のSビットがセットされていると，そのファイルを「シングル・ユーザ・モード」でオープンします．この排他処理(検査)にも，やはりパス・デスクリプタのリンク・リストが使用されますが，チェックされるのは，RBF独自のPD_Confl(後の説明を参照)によるリス

コラム3

パス・リスト(pathlist)の解析

▶パス・リスト→ファイル・マネージャ

I$Create，I$Open，I$MakDir，I$ChgDir，I$-Deleteの各システム・コールではパス・リストを扱いますが，カーネルはどのようにしてパス・リストから適切なファイル・マネージャ(とデバイス・ドライバ)を呼び出すのでしょうか．その手順はパス・リストが/で始まっているかどうか，すなわち絶対パス・リストの場合と相対パス・リストの場合とで多少異なります．

(1) 絶対パス・リストの場合

絶対パス・リストの先頭の要素(/の直後)は，デバイス名(デバイス・デスクリプタのモジュール名)です．カーネルは，デバイス・テーブルのエントリを得るために，このデバイス名を使って内部的にI$Attachを実行します．

I$Attachは，デバイス名のデバイス・デスクリプタをF$Linkでリンクし，もしそのデバイス名のデバイスがすでにデバイス・テーブルに登録されているならば，そのユース・カウントを増やし，そこに登録されているファイル・マネージャとデバイス・ドライバ・モジュールのリンク・カウントも増やします．

もしそのデバイスがまだデバイス・テーブルに登録されていない場合は，デバイス・デスクリプタ・モジュール中に書かれているファイル・マネージャとデバイス・ドライバ・モジュールもリンクして，とりあえずデバイス・テーブルに登録します．さらにデバイス・テーブルを走査して，もし登録したデバイスと物理的に同じデバイス(ポート・アドレスとデバイス・ドライバが同じデバイス)がまだ登録されていなければ，デバイス・ドライバのM$Memに書かれているサイズのデバイス・スタティック・ストレージを確保してデバイス・テーブルに登録し，デバイス・ドライバの初期化ルーチン(INIT)を呼んでデバイスとスタティック・ストレージを初期化します．すでに物理的に同じデバイスが登録されていれば，新たにスタティック・ストレージを確保したりはせずに，たんにアドレスを新しいエントリにコピーするだけです．

いずれの場合も，I$Attachで得られたデバイス・テーブルのエントリ・アドレスをパス・デスクリプタのPD_DEVフィールドに書き込み，ファイル・マネージャを呼び出します．

(2) 相対パス・リストの場合

OS-9の各プロセスは，カレント・データ・ディレクトリとカレント実行ディレクトリの二つのデフォルト・ディレクトリを必ずもっています．これらのデフォルト・ディレクトリの情報は，プロセス・デスクリプタのP$DIOフィールドに書かれています(前半がカレント・データ・ディレクトリで後半がカレント実行ディレクトリ)．P$DIOには，カレント・データ・ディレクトリとカレント実行ディレクトリにそれぞれ16バイトずつ，計32バイトぶんのフィールドがありますが，カーネルが使用するのはそれぞれの先頭の4バイトで，デフォルト・デバイスのデバイス・テーブル・エントリ・アドレスが書き込まれます．

カーネルは，相対パス・リストを処理するときには，まずアクセス・モード(R$d0.b)からカレント・データ・ディレクトリとカレント実行ディレクトリのどちらを使用するのかを決め，システム・コールを発行したプロセスのプロセス・デスクリプタのP$DIOから当該デフォルト・デバイスのデバイス・テーブル・エントリ・アドレスを得て，必要なファイル・マネージャを呼び出します．

P$DIOフィールドの残りの部分の使用は，デフォルト・デバイスのファイル・マネージャにまかされており，RBFではデフォルト・ディレクトリのファイル・デスクリプタ・セクタの論理セクタ番号(LSN)が，NFMでは，さらに相手ステーションのステーションIDと相手

トです．この方法による排他制御は，レコード・ロッキングの特別な場合です．

◇**バッファの確保**

ファイル・マネージャによっては，パスに固有の私的なバッファを必要とするものがありますので，F$SRqMem で確保し，そのアドレスをパス・デスクリプタに記録します．

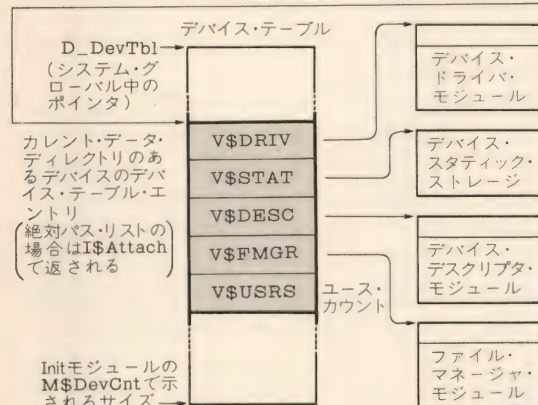
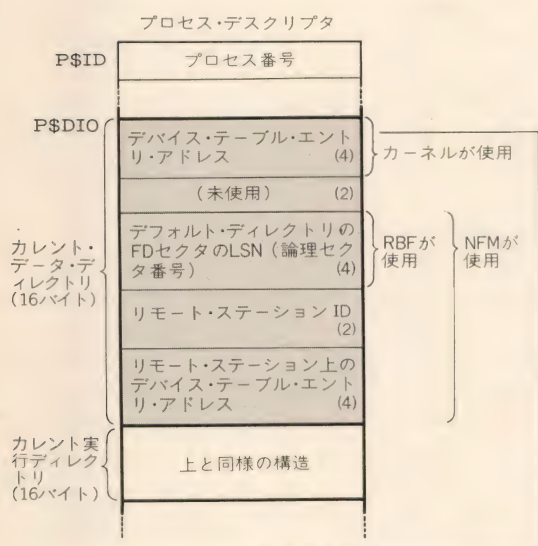
SCF　行編集用バッファ
RBF　セクタ・バッファ
　　(ビットマップ用，FD セクタ用，データ用)
Pipe　FIFO バッファ(デフォルト以外のバッファ・サイズが指定された場合)

◇**パス・デスクリプタのリスト構造**

カーネルは，同一のデバイス(ポート・アドレスが同じでデバイス・ドライバも同じデバイス)にオープンされたすべてのパスのパス・デスクリプタをリスト構造で結合します．

V_Paths → PD_Paths → PD_Paths…

上記のリストはすべてのファイル・デバイスに共通ですが，ファイル・マネージャによっては，上記のリスト構造に加えて，そのファイル・マネージャ独自のリスト構造でパス・デスクリプタを結合する場合もあ

〔図A〕デフォルト・ディレクトリの検索

ステーションにおけるデバイス・テーブル・エントリ・アドレスが書かれています(図A)．

このようにデバイス・テーブルは，そのデバイスの属性を表すうえで重要なデータ構造です．デバイス・テーブルのベース・アドレスはシステム・グローバルの D_DevTbl に，サイズ(登録できるデバイスの最大数)は init モジュール(D_Init)の M$DevCnt に書かれています．

▶**パス番号→パス・デスクリプタ**

カーネルが行うもう一つのマジックは，パス番号からパス・デスクリプタを得ることです．これには，二つの段階があります．デバイス・ドライバやファイル・マネージャでない通常のユーザ・モード・プログラム(OS-9 に付属のユーティリティやコンパイラなども含む)で扱うパス番号は，正確には「ユーザ・パス番号」で，そのプロセス内でユニークな 0 ～31 の整数です．カーネルは，ユーザ・プログラムから I$Read などのシステム・コールが発行されると，そのパス番号(R$d0.w)をプロセス・デスクリプタ中の「パス・テーブル」(P$Path)を使用して，「システム・パス番号」に写像します．システム・パス番号はシステム全体でユニークな 1 以上の番号で，「パス・デスクリプタ・テーブル」のインデックスです．

パス・デスクリプタ・テーブルとは，システム中に存在するすべてのパス・デスクリプタへのポインタの配列で，そのベース・アドレスはシステム・グローバル中の D_PthDBT で指されます．パス・デスクリプタ・テーブルの先頭(0 番目)の要素は，現在割り当てられているパス・デスクリプタ・テーブルのサイズと各パス・デスクリプタそのもののサイズ(現在のバージョンでは 256 バイト)が書かれています(いずれもワード)．システム・モード・パス番号が 1 から始まるのはこのためです．また，(そのパスがクローズされたなどで)実際には使用されていないスロットには $0 が書かれています．

全プロセス・デスクリプタへのポインタの配列であるプロセス・デスクリプタ・テーブルもほぼ同じ構造をし

ります.

(1) RBF の場合

相互排除のために，以下のような一つのリスト構造が存在します.

① 同一ドライブにオープンされたパスのリスト(ビットマップの更新制御のため)

V_FileHd → PD_NiFil → PD_NxFil…

② 同一ファイルにオープンされたパスのリスト(レコード・ロッキングのため)

PD_Confl → PD_Confl → PD_Confl…

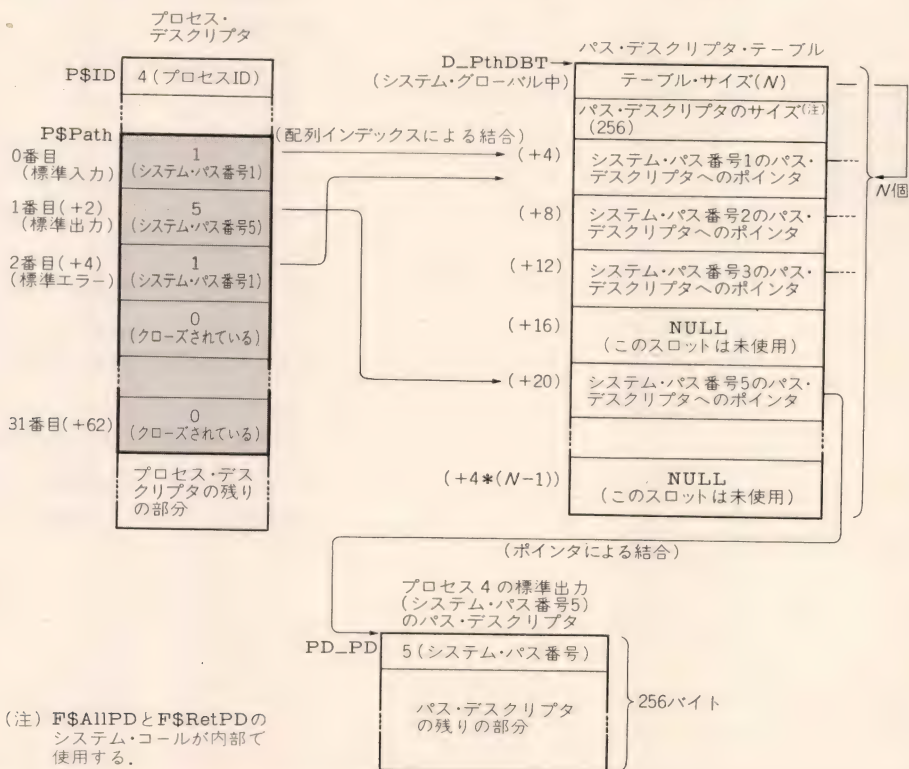
上記のリストの作成では，まずドライブ・テーブル(スタティック・ストレージの一部)のアドレスを比較してパスを同一ドライブに属するパスのリストに入れ，ついでファイル・デスクリプタ・セクタの論理セクタ番号を比較して同一ファイルに属するパスのリストに入れます(図12).

(2) Pipe の場合

ある「パイプ・デバイス」にオープンされたすべてのネームド・パイプのパス・デスクリプタは，スタティック・ストレージから始まる双方向のリスト構造(循環リスト)で連結されています./pipe をディレクトリ・モードでオープンすると，パイプ・ファイル・マ

ています(ただしプロセス・デスクリプタのサイズは現バージョンでは 512 バイト；D_PrcSz に書かれている). これらのテーブルは，その初期サイズが init モジュールのそれぞれ M$Paths と M$Procs に書かれていますが，パス(またはプロセス)が増えてサイズが足りなくなると，自動的に拡張されます.

パスのオープンはユーザ・モードだけとはかぎりません. たとえば，デバイス・ドライバ中からフォント(字形)ファイルをオープンしたり，OS-9 ver 2.0 からサポートされた「システム・ステート・プロセス」(カーネル，ファイル・マネージャ，デバイス・ドライバなどのようにシステム・ステートで動作するプロセス)を使用する場合などです. システム・モードのプログラムが扱うパス番号はシステム全体にユニークなシステム・モード・パス番号のほうですから，注意が必要です. すなわち，Unix や OS-9 の解説書にある「パス番号の 0 は標準入力，1 は標準出力，…」は，もはや通用しません. 文献 10)にシステム・ステート・プロセスで標準入出力パスのバッファード入出力関数(printf( )など)を使用する方法を示してありますので参照してください(図B).

〔図B〕
ユーザ・パス番号とシステム・パス番号/パス・デスクリプタの関係

ネージャはこのリスト構造を使用して，疑似的なディレクトリ・エントリを読み出せるようにします(図13)．

◇SS_Open

ファイル・マネージャで行うべき処理がすべて終了したら，SS_Openの機能コードをもってデバイス・ドライバのPUTSTAのエントリを実行します．この慣習はver 2.0のRBFやSCFから追加され，デバイス・ドライバもファイルがオープンされたことを知ることができます．RBFやSCFのデバイス・ドライバでは，ファイルがオープンされても通常とくに何もすることがありませんから，E$UnkSvc(該当する機能をサポートしていない)エラーを返しますが，ファイル・マネージャはこのエラーを無視します(カーネルには「エラーなし」でもどる)．

後に紹介するVSFファイル・マネージャのように，ファイル・マネージャはほとんどダミーで，オープン時のパス・リストの解析を含めてほとんどの処理をデバイス・ドライバで行う場合は，このSS_Openが役立ちます．

〔図12〕RBFのパス・デスクリプタ・リスト構造

▶ MakDir(I$MakDir)

このエントリをサポートするのは，階層ディレクトリ構造をもつ書込み可能なマルチファイル・デバイスだけです．

◇処　理

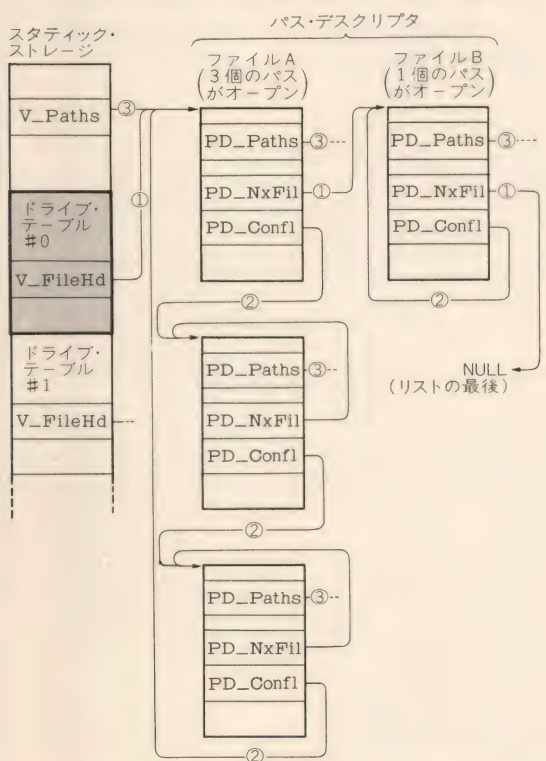
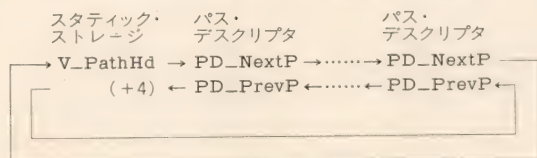
ほとんどの部分がCreateのエントリと共通になります．RBFの例では，Createを内部でサブルーチンとして呼び出した後，ファイル内容の初期化("."と".."のエントリの書込み)と，ファイル・アトリビュート(Dirビット)の変更を行っているだけです．

▶ ChgDir(I$ChgDir)

このエントリをサポートするのは，マルチファイル・デバイスだけですが，すべてのマルチファイル・デバイスがサポートしているわけではありません．たとえばPipeは名前付きパイプをサポートするマルチファイル・デバイスですが，ChgDirはサポートしていません．同様に，NRF(Non-volatile RAM File Manager)でもサポートされていません．

◇処　理

そのディレクトリにカレント・ディレクトリを設定できるということは，そのディレクトリが存在していて自分にアクセス権がある，すなわちオープンできるということです．与えられたパス・リストに対してディレクトリ・モードでOpenの処理を内部で実行して，得られた情報のうちカレント・ディレクトリを特定するために必要なもの(RBFの場合であれば，デフォルト・ディレクトリのFDセクタのLSN)をプロセス・デスクリプタのP$DIOに格納します．

I$ChgDirのアクセス・モード(R$d0.b)のWビットがセットされている場合，RBFの通常のOpenの処理では最終変更日時を更新するためのディスクへの書込みが生じ，メディアが書込みプロテクトされている

〔図13〕名前付きパイプの循環リスト構造

とエラーになりますが，ChgDir の場合にかぎり，物理的な書込みが失敗してもエラーとはみなさないようになっています．

▶ Delete(I$Delete)

このエントリをサポートするのは，書込み可能なマルチファイル・デバイスだけです．実際にサポートしているのは RBF，Pipe，NFM などです．

◇処　理

まず，そのファイルが存在して，Delete を実行するプロセスが書込み許可をもっているかどうか，すなわち書込みモードでオープンできるかどうかを検査します．存在しないファイルを削除しようとするのはエラーです．つぎに，そのファイルが占有していた記憶領域を返却し，最後にそのファイルを示すデータ構造を削除します．

(1) RBF の場合

削除するファイルがディレクトリ・ファイルの場合は，そのディレクトリが空("." と ".." 以外のエントリがない)であることを確認します(空でないときは E$DNE エラー)．ただし，ルート・ディレクトリを削除することはできません．

ビット・マップ・セクタを操作して，ファイルの実体と FD セクタを返却し，そのファイル(の FD セクタ)を指しているディレクトリ・エントリを削除します(ファイル名の先頭バイトを $00 にする)．

(2) Pipe(名前付きパイプ)の場合

拡張バッファが使用されていれば F$SRtMem でメモリを返却します．パス・デスクリプタをリストからはずし，F$RetPD で削除します．

◇オープンしているファイルの削除

RBF や Pipe では，オープンしているファイルを削除しようとすると E$Share エラーを返し，そのファイルは削除されません．これは，いくつかある Unix との細かな違いの一つですから，Unix のプログラムを移植するときには注意が必要です．Unix のファイル・システムでは，ファイルのマルチリンクを許しており，unlink( )システム・コールはリンク・カウントを一つ減らします．unlink( )でリンク・カウントが 0 になっても，ただちにファイルが削除されるわけではなく，そのファイルがオープンされているかぎりファイルの実体は存在しますから，以下のようなワザでプロセスが終了(exit( ))すると自動的に削除される一時ファイルを使用できますが，OS-9 ではこの方法は通用しません．

```
fd = creat("temp", MODE);
unlink("temp");
```

RBF のファイル・デスクリプタ・セクタにも，一応 FD_LNK というフィールドが予約されていますが，Create 時に，1 にセットされ，Delete 時に 0 に減少されるだけで，既存のファイルを別のディレクトリにリンクするシステム・コール(link( ))はありませんから，実際にはサポートされていないのと同じです．

削除するファイルがオープンされているかどうかの検査には，ファイル・マネージャ独自の「同一ファイルにオープンされたパスのリスト」(Open の項を参照)を使用します．

▶ Seek(I$Seek)

ディスクなどのランダム・アクセス可能なデバイスだけがサポートし，ファイル上の論理位置を移動します(物理ヘッドの移動は行わない)．書込みモードのときは，もしまだ書き込まれていないブロック(セクタ)・バッファがあれば，その内容をディスクに書き出します．もし現在のファイル・サイズを超える位置に移動するときは，そのぶんの記憶領域を確保し，そのぶんファイル・サイズを拡張します．読出しだけのモードのときは，現在のファイル・サイズを超えてシークすることはできません．

キャラクタ・デバイスやテープなど，ランダム・アクセスができないデバイスのファイル・マネージャは Seek のエントリをサポートしませんが，エラーにはせずに何もしないで(キャリをクリアして)カーネルに戻ります．その理由は，言語(Pascal など)によっては，プログラムの実行開始時にオープンされているファイルを必ずリワインド(ファイルの先頭にシーク)するものがあるからです．

▶ Read(I$Read)

デバイス・ドライバの READ エントリを呼び出して，データをユーザ・バッファに読み込みます．デバイスの特性により，以下のような場合があります．

(1) **キャラクタ・デバイス**(SCF，Pipe)

デバイス・ドライバからデータを 1 文字(バイト)ずつ，ユーザ・バッファが一杯になるまで読み込んでいくだけです．データの加工や解釈(行末文字や EOF 文

字)の解釈はいっさいありません.

Pipeでは，デバイス・ドライバ(null)はダミーで，実際に呼び出されることはなく，ファイル・マネージャがすべてを処理します．パイプ・バッファ中に指定されただけの量のデータがないときには，必要な量のデータが(別プロセスにより)書き込まれるまで，読出しプロセスはスリープ状態になります.

(2) 可変長ブロック・デバイス(IBF，SBBF)

ファイル・マネージャは，デバイス・ドライバにユーザ・バッファのアドレスとサイズを渡すだけで，ほとんど何もしません．デバイス・ドライバはユーザ・バッファが一杯になるまでデータを読み込みます．バッファリングするかどうかは，デバイス・ドライバによります.

IBF(IEEE488/GPIB用ファイル・マネージャ)のデバイス・ドライバでは，ENDメッセージ(EOIライン)を検出すると，これを絶対的な転送レコードの終結(EOR)として扱いますので，実際の転送サイズがユーザの指定したバッファ・サイズより小さくなることがあります．また，IBFは，バス上のデバイスが指定されているときには，実際のデータ転送に先立って，トーカ/リスナ指定のためのインターフェース・メッセージ(ATN＝True)を送信します．この判断はファイル・マネージャが行い，デバイス・ドライバのSS_SCmdのPUTSTAを呼び出します.

(3) 固定長ブロック・デバイス(RBF，SBF)

デバイス・ドライバは，固定サイズのブロック(セクタ)単位で入出力しますから，パスをオープンするときにあらかじめ1ブロックぶんのバッファを確保しておきます．もし読出し開始位置がブロック・サイズの整数倍でないならば，いったんブロック・バッファに読み込んでから，必要な位置からユーザ・バッファにコピーします．もし読出しが前回の読出しのつづきならば，前回読み込んだブロックがまだ使えますから，デバイス・ドライバは呼び出しません．読出しがブロックの途中で終わるときは，やはりいったんブロック・バッファに読み込んでから，必要なバイト数をユーザ・バッファにコピーします.

ver 2.0のRBFからは，デバイス・ドライバは「可能なかぎり」ユーザ・バッファに直接転送することになりましたから，まるごと読み込まれるセクタは，ブロック・バッファを経由しないで直接ユーザ・バッファに転送されるものと思われます(**図14**).

〔図14〕 RBFのセクタ読込み

ファイルの終わりを越えて読み込むことはできません．ファイルの読出し位置が現在のファイル・サイズを超える，すなわちEnd-of-Fileが検出されたら，実際の読出しサイズを現在のファイル・サイズまでとします．読出し開始位置がすでにファイルの最後にあるならば，E$EOFエラーを返します.

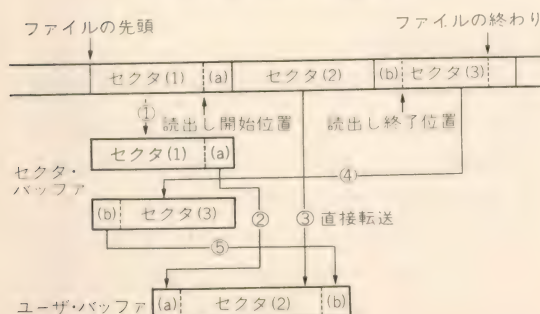
▶ **Readln**(I$ReadLn)

SCFのような対話型デバイスを扱う場合は，行編集を行わなければなりません．このために，SCFではパスがオープンされたときに行編集用のバッファ(256バイト)を確保し，この中で1行ぶん(キャリッジ・リターンまで)を編集してからユーザ・バッファにコピーします．ですからSCFでは，I$ReadLnで一度に読み込めるのは最大256バイトまでで，しかもキャリッジ・リターンが入力される前に行編集バッファが一杯になっても，ベル(PD_OVF)を鳴らして入力をつづけます(キャリッジ・リターンが入力されるのを待つ).

対話操作を行わないデバイスでは，ユーザ・バッファが一杯になるか，キャリッジ・リターンが入力されるか，またはEnd-of-Fileが検出されるかのいずれか早いところまで入力すること以外，Readの動作と変わりありません．ただし，IBFでは入力(接続される機器)の行末のデリミタ文字にバリエーションがありますので，以下のような変換処理をします.

<IBFのReadLnの行末文字処理>

キャリッジ・リターンだけ　(そのまま)

入力	処理
ライン・フィード だけ (Unix風)	キャリッジ・リターン (OS-9の行末文字)に変換
キャリッジ・リターン＋ライン・フィード (CP/M，MS-DOS風)	

▶ Write (I$Write)

転送方向が逆なこと以外，Read の処理とほとんど同様の動作をします．RBF のような固定長ブロックのランダム・アクセス・デバイスでは，書込みがブロック(セクタ)の途中から始まる場合は，いったん現在のファイルの内容をセクタ・バッファに読み出してから内容を書き換えて書き込みます．現在のファイル・サイズを超えて書き込むと，ファイル・サイズが自動的に拡張されます．

(1) Pipe の場合

パイプ・バッファ中に指定されただけの量のデータをすべて書き込めるぶんの空きがないときは，別プロセスによりデータが読み出されて空きができるまで，書込みプロセスはスリープ状態でブロックされます．

(2) IBF の場合

バス上のデバイスが指定されているときには，実際のデータ転送に先立って，トーカ/リスナ指定のためのインターフェース・メッセージを送信します．

▶ Writeln (I$WritLn)

Write と Writeln には，Read と Readln ほどの違いはありません．まず最初に，バッファ中にキャリッジ・リターンがあれば，転送するデータがそこまでになるようにバッファ・サイズ(R$d1.l)を調節します．後の処理は Write の場合とほとんど同じです．

SCF や IBF では，パス・デスクリプタのオート・ライン・フィード・フラグがセットされていて，もし最後に出力したキャラクタがキャリッジ・リターンの場合は，もう 1 バイト，ライン・フィードを出力しますが，SCF ではこの処理(判断)をファイル・マネージャが行い，IBF ではデバイス・ドライバが行っています．これは，SCF はもともと文字単位でデバイス・ドライバを呼び出し，IBF ではブロック単位で呼び出しているからです．この付加されたライン・フィードは，ユーザに返される実際に出力されたバイト数(R$d1.l)には含めません．

余談ですが，OS-9 が行編集ありの入出力(ReadLn/WritLn)と行編集なしの入出力(Read/Write)とをはっきり分離して，別々のシステム・コールで扱うようにしてあるのは，Unix や MS-DOS に比べて，明らかに優れていると思います．これにより，同じ read( ) のシステム・コールを発行してもターミナルとディスクとでは行末文字を処理する/しないが違うという「マジック」がなくなり，より統一化された入出力システムになったといえます．

もちろん，Unix でも fstat( ) システム・コールで種々のモードを切り替えることができますが，「モード切替え方式」は，現在どちらのモードなのかを記憶または問い合わせなければならず，read( )/readln( ) 方式に比べると，明らかに使いやすさの点で劣ります．

▶ GetStat (I$GetStt)

ファイル・マネージャで処理する代表的な GetStat 機能を以下に示します．これ以外の機能コード(SS_***)が渡されたときには，そのままデバイス・ドライバの GETSTA を呼び出します．

● SS_Opt (全デバイス)

パス・デスクリプタのオプション部分(後半の 128 バイト)をユーザ・バッファにコピーします．全ファイル・マネージャに共通ですから，カーネルが処理してもよさそうですが，逆手にとってオプション部分の実際の内容を加工して返すこともできます．

● SS_Ready (SCF，IBF，Pipe などのシーケンシャル・アクセス・デバイス)

デバイス・ドライバの入力バッファ中にデータ・バイトがすでに受信されているかどうかを検査し，もしあればそのバイト数を，なければ E$NotRdy エラーを返します．ランダム・アクセス・デバイスではつねに「1 バイトある」と返します．

● SS_Size (RBF のようなランダム・アクセス・デバイスと Pipe)

現在のファイル・サイズを返します．Pipe ではバッファ・サイズを返します．

● SS_Pos (RBF のようなランダム・アクセス・デバイス)

現在のファイル・ポインタ(つぎの Read/Write の開始点)の位置をファイル先頭からのバイト数で返します．C 言語のライブラリ関数 lseek( ) は，I$Seek を実行してからこの GetStat を実行して，現在のファイル・ポジションを返します．

● SS_EOF (RBF のようなランダム・アクセス・デバイスと Pipe)

現在のファイル・ポインタがファイルの最後ならば E$EOF エラーを返し，そうでなければエラーなしで戻ります．シーケンシャル・アクセス・デバイスではつねにエラーなしで戻ります．

● SS_FD(RBF や Pipe のようなディレクトリ構造をもつデバイス)

FD セクタの内容をユーザ・バッファにコピーします．DIR のようなユーティリティ・プログラムで使用できるようにするため，RBF 以外のデバイスでも，RBF の FD セクタ(先頭の 16 バイト)の形式の疑似データを返します．

▶ SetStat(I$SetStt)

ファイル・マネージャで処理する代表的な SetStat の機能を以下に示します．これ以外の機能コード(SS_***)が渡されたときには，そのままデバイス・ドライバの PUTSTA を呼び出します．

● SS_Opt(全デバイス)

パス・デスクリプタのオプション部分(後半の 128 バイト)をユーザ・バッファからコピーしますが，全部コピーする必要はなく，必要または可能なものだけコピーします．まったくコピーしなくてもかまいません．

● SS_Size(RBF のようなランダム・アクセス・デバイスと Pipe)

ファイル・サイズを指定された値に変更します．指定されたサイズが現在のファイル・サイズより小さい場合にはファイルを切り捨て，現在のファイル・サイズより大きい場合にはファイルを拡張します．一般的には，拡張された部分の内容は不定，すなわち記憶領域を確保するだけで，実際には何も書き込みません．

● SS_Reset(RBF，SBF，IBF)

RBF と SBF ではデバイス・ドライバで処理され，物理的なヘッド位置(またはメディア位置)をホーム・ポジションに戻します．IBF ではデバイス・クリアを発行します．

● SS_FD(RBF や Pipe のようなディレクトリ構造をもつデバイス)

COPY コマンドでファイルをコピーするときに，元のファイルのオーナ，作成・修正日時などを保存するために使用されます．RBF 以外のデバイスでも RBF の FD セクタ(先頭の 16 バイト)の形式の疑似データを処理するようにします．ただし，FD_Attr(ファイル・アトリビュート)とサイズの情報はこの SetStat で設定できません．

● SS_Ticks(RBF，IBF)

RBF では，レコード・ロッキングで待たされるプログラムがスリープする最大ティック数を設定します．IBF では，デバイス・ドライバが実際の入出力処理でスリープする最大時間を設定するために使用しています．一般的なタイムアウトの設定に使用してもよいでしょう．

● SS_SSig(SCF，IBF，Pipe などのシーケンシャル・アクセス・デバイス)

データ・バイトが受信されたときに送るシグナルを登録します．SCF ではデバイス・ドライバが処理し，シグナル・コードとプロセス ID をデバイス・スタティック・ストレージに登録し，Pipe ではパス・デスクリプタに登録していますが，Pipe の場合はパス・デスクリプタ＝ストレージ・デバイスですから同じことです．IBF では，シグナル・コードをパス・デスクリプタに登録し，そのパスを最後に使用したプロセスにシグナルを送ります．

● SS_Relea(SCF，IBF，Pipe などのシーケンシャル・アクセス・デバイス)

SS_SSig で設定したシグナルの登録を取り消すなど，内部の設定の解除に使用します．IBF ではタイムアウトのリセット(無限大時間の待ち)にも使用しています．

● SS_Attr(RBF や Pipe などファイルがアトリビュートをもつデバイス)

ディレクトリ・ファイルのアトリビュートを通常ファイルに変更するときには，いくつかの条件があります．すなわち，そのディレクトリ中の(自分と親を除く)すべてのエントリが空でなければなりません．また，ルート・ディレクトリを通常ファイルに変更したり，通常ファイルをディレクトリに変更することはできません．

▶ Close(I$Close)

Close の処理は，Open に比べるとはるかに簡単です．要するに，そのパスを閉じるための始末をして，痕跡を残さないようにすればよいのです．

① 固定ブロック・モードのデバイス(RBF，SBF)の場合，書込みモードでまだメディアに書き込まれていないバッファ(ダーティ・バッファ)があれば，メディアに書き出す．

② パス・カウント(PD_CNT または PD_COUNT)を見て，そのパスがまだ使われているなら，それ以上何もしないで戻る．これは，カーネルが I$Dup で複製されたパスに I$Close が発行されるたびに(パ

ス・カウントを減少させて)ファイル・マネージャを呼び出すため.

③ ファイルの属性などをメディアに記録する必要があるならば記録する.

④ デバイス・ドライバのSS_CloseのPUTSTAを実行して，デバイス・ドライバにもパスをクローズすることを伝える．SS_Openの場合と同じく，デバイス・ドライバがSS_CloseのPUTSTAをサポートしてなく，E$UnkSvcエラーを返してきても無視する.

⑤ ファイル・マネージャ独自のリスト構造からそのパスを取り除く(RBFとPipe).

⑥ Openのときに確保した私的なバッファがあれば，その領域をF$SRtMemでシステムに返却する.

以上でファイル・マネージャの処理は終わりです．パス・デスクリプタの解放やデバイスに対するリンク・リストの処理，プロセス・デスクリプタのパス番号の処理などはカーネルが行います.

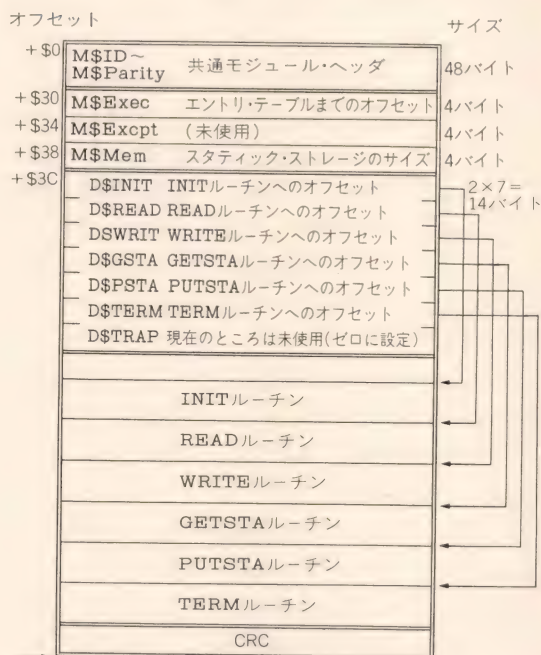
ただし，Pipeの場合は多少動作が違います．クローズされるパイプが名前なしの場合は，たとえバッファにまだデータが残っていてもその内容は捨てられてしまい，パイプは消滅してしまいますが，名前付きパイプでバッファ中にデータが残っている場合は，ファイル・マネージャがパス・カウント(PD_CNTとPD_COUNT)を1だけ増加させて，カーネルがそのパス・デスクリプタを削除してしまわないようにします．というのは，パイプではパス・デスクリプタがファイルそのものなので，パス・デスクリプタを削除することはファイルを削除(Delete)することを意味するからです．このため，名前付きパイプでは，そのファイル(パス)をだれもオープンしていないのにパス・デスクリプタが存在するという，一見奇妙な状態になりますが，こうしないと下のような名前付きパイプ特有の操作ができなくなります.

```
$ echo hello >/pipe/junk
$ list /pipe/junk
hello
$ ■                (下線部がユーザの入力)
```

名前付きパイプがクローズされても残るのは，バッファ中にまだデータがある場合だけですから,

```
$ touch /pipe/abc
```

として空の名前付きパイプを作成しても残りません.

〔図15〕デバイス・ドライバの一般的な構造
(各ルーチンの実際の位置は図のとおりでなくてもよい)

4 デバイス・ドライバとのインターフェース

ファイル・マネージャは，実際にハードウェアを操作したり，ハードウェアに依存する操作などを行うために，必要に応じてデバイス・ドライバを呼び出します．以下に，ファイル・マネージャとデバイス・ドライバのインターフェースの方法を示します.

4.1 エントリ

OS-9のデバイス・ドライバには，INIT, READ, WRITE, GETSTA, PUTSTA, TERM, TRAPの7個の標準エントリがあります．このうち，INITとTERMはカーネルから直接実行され，ファイル・マネージャから呼び出されることはありません．また，TRAPは，現在のところ予約されているだけで実際には使用されていませんから，ファイル・マネージャが呼び出すのは，READ, WRITE, GETSTA, PUTSTAの4個だけです.

デバイス・ドライバの各エントリは，デバイス・ド

ライバのモジュール・ヘッダ中の M$Exec で指されるオフセット・テーブルに，モジュールの先頭からそのエントリまでのオフセットが書かれています(図15)．ファイル・マネージャのエントリ・オフセット・テーブルの内容は，オフセット・テーブルの先頭からそのエントリまでのオフセットでしたが，この違いの理由はわかりません．OS-9/6809 の時代は lbra 命令によるジャンプ・テーブルでしたから，「趣味の問題」程度のことなのかもしれません．ただし，実際にはファイル・マネージャの方式のほうが，モジュール・ポインタが不要なぶんだけ呼出しプログラムがほんの少し楽になります．

オフセット・テーブル中の各オフセットの位置は，D$INIT～D$TRAP のシンボルで定義されています．**リスト 1** にデバイス・ドライバの呼出しディスパッチ・ルーチンの例を示します．

ファイル・マネージャの各処理ルーチンがどのようにデバイス・ドライバを呼び出すかは，まったくファイル・マネージャの自由です．したがってデバイス・ドライバの各エントリの処理内容も，ファイル・マネージャとデバイス・ドライバ間で約束しておけば，何をやってもかまいません．

たとえば，IBF(IEEE488/GPIB ファイル・マネージャ)のデバイス・ドライバでは，デバイス・ディペンデント・メッセージ(データ)は READ/WRITE のエントリで処理しますが，インターフェース・メッセージ(トーカ/リスナの指定など)は PUTSTA のエントリで処理します．したがって，IBF の Read のエントリでは，まずデバイス・ドライバの PUTSTA を実行してトーカとリスナの指定をしてから，READ を実行して実際にデバイス・ディペンデント・メッセージを読み込みます．

〔リスト 1〕デバイス・ドライバの呼出しディスパッチ・ルーチンの例

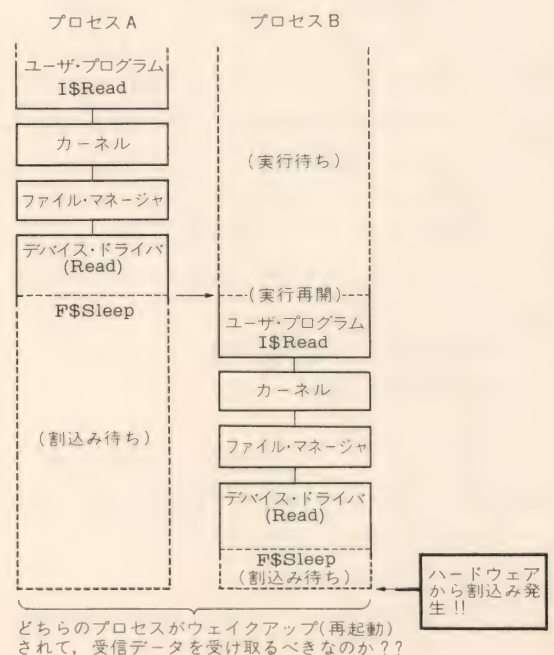
```
* calldrvr  デバイス・ドライバの呼出し
* 入力 (a1) =  パス・デスクリプタのアドレス
*      (a6) =  システム・グローバル・ベースのアドレス
*      d7.l =  オフセット・テーブル・インデックス
*              D$READ,D$WRIT,D$GSTA,D$PSTA
*      その他のレジスタは必要に応じて設定しておく
* 破壊 d0-d7/a0-a6(デバイス・ドライバからの戻り値を含む)
*      破壊されて困るレジスタはスタックに退避しておく
calldrvr: movea.l PD_DEV(a1),a3   デバイス・テーブル・エントリ
          movea.l V$STAT(a3),a2   スタティック・ストレージ
          movea.l V$DRIV(a3),a3   デバイス・ドライバ・モジュール
          add.l M$Exec(a3),d7
          move.w (a3,d7.l),d7     オフセットを得る
          jmp (a3,d7.w)           エントリにジャンプ
```

また，GKSMan では標準の 7 個のエントリに加えて，全部で 39 個のエントリのドライバを使用します．

しかし，いくらファイル・マネージャとデバイス・ドライバの間で決めれば何をやってもよいとはいえ，デバイス・ドライバの READ を呼び出すとドアが開いてメディアが飛び出すような設計がよいかどうかは設計者の常識の問題でしょう．

4.2 相互排除

すでに強調したように，デバイス・ドライバはハードウェア・ポートとそれに密接に関連したスタティック・ストレージという共有不可能な資源を操作しますので，一般的にはリエントラント(再入可能)ではありません(一つのデバイス・ドライバで物理的に別のデバイス・ポートを並行して扱うことはできる．この場合は，スタティック・ストレージもポートごとに独立している)．したがって，ファイル・マネージャは，あるデバイス・ポートを扱うデバイス・ドライバをただ一つのプロセスしか実行しないように，相互排除処理を

〔図16〕マルチタスキングで入出力の割込みを待つプロセスどうしの衝突(相互排除なし)

行わなければなりません．相互排除処理はデバイス・ドライバで行ってもよいのですが，OS-9 全体の約束(または慣習)として，ファイル・マネージャ内で行うことになっています．

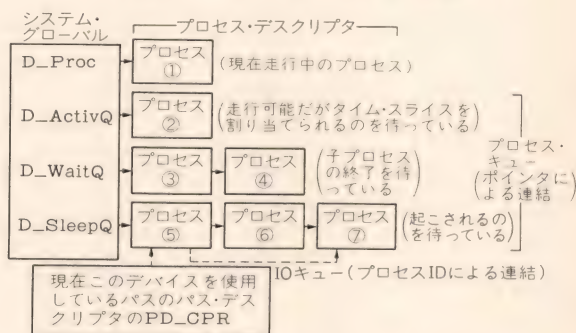
デバイスの使用に対するプロセスの衝突は，割込みによって生じます．すなわち OS-9 では，ユーザ・プロセスから発行されたシステム・コール(もちろん入出力シスム・コールを含む)を実行中などで，プロセスがシステム・ステートで走行中(Unix でいうところのカーネル・モード)は，プロセス切替えを行いません．したがって通常に命令を実行しているかぎりは，一度デバイス・ドライバを実行し始めたプロセスは，タイマ割込みなどが起こっても割込みから復帰後はそのまま実行をつづけ，他のプロセスが走行することはありませんから，相互排除の問題を考える必要はありません．

しかし，現実にはハードウェアが使用可能になるまで，ハードウェアに動作を命令してから実行が完了するまで，外部機器からデータが到着するまでなど，何かの事象が発生するのを待たねばなりません．事象の発生を待つためには，プロセスをスリープ状態にしてハードウェアからの割込みを待ち，割込み処理ルーチン内でその事象を待っているプロセスを起こさなければなりません(スリープ状態からアクティブ状態にする)．このように，ハードウェアの動作を待つプロセスが F$Sleep を発行すると，カーネルはアクティブ・キューからつぎのプロセスを取り出してタイム・スライスを与え，プロセス切替えが起こります(図16)．

入出力の排他制御のためには，F$IOQu システム・コールを使用します．F$IOQu には，使用したいデバイスを現在占有しているプロセス番号をパラメータとして与えます．F$IOQu を実行したプロセスはスリープ状態になります(スリープ・キューに入る)が，同時にそのデバイスの IO キュー(Queue, 待ち行列)にも入ります(図17)．スリープ・キューはプロセス・デスクリプタのポインタのリストによるキュー(P$QueueN と P$QueueP)ですが，IO キューはプロセス番号によるキュー(P$IOQN と P$IOQP)です(図18)．

現在デバイスを使用しているプロセスは，そのプロセス番号がデバイス・スタティック・ストレージの V_BUSY フィールドに書かれています．この書込みはファイル・マネージャが行い，使用が終了したらファイル・マネージャが必ず V_BUSY をクリアしなければなりません．デバイスを使用していたプロセスがデバイスの使用を終了し，デバイス・ドライバ→ファイル・マネージャ→カーネルと復帰すると，カーネルはそのプロセスから始まる IO キューに属する(P$IOQN と

〔図17〕キューイングによる入出力デバイスの衝突の回避
(相互排除あり)

〔図18〕プロセス・キューと I/O キュー
(実際のプロセスのキューは双方向の循環リスト)

プロセス⑤はデバイスの動作が完了(割込みを発生)するのを待っている．
プロセス⑦はプロセス⑤がデバイスの使用を終了するのを待っている．

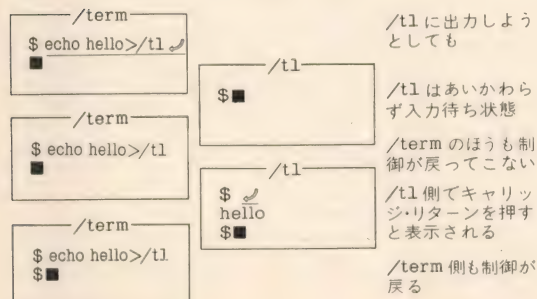
P$IOQP で連結されている)プロセスをすべて起こし(スリープ・キューからアクティブ・キューに移し)ますから，F$IOQu を発行したプロセスはシステム・コールから復帰します．IO キューから目覚めたプロセスは，デバイスを使用できるかどうか再びチェックします．

リスト 2 にデバイスが使用可能かどうかを検査するルーチンの例を示します．実際には，このルーチンは**リスト 1** のデバイス・ドライバの呼出しディスパッチ・ルーチンと組み合わされた形でファイル・マネージャに組み込まれます．後出の**リスト 3** の VSF の例を参照してください．

上記の F$IOQu によるデバイス・ドライバの呼出し時の排他制御は，あくまでも一般論です．ファイル・

〔リスト 2〕デバイスが使用可能かどうかのチェック・ルーチンの例

```
* chkdev  デバイスが使用可能かどうかチェックする
*         使用可能なら占有する
* 入力 (a2) =  ディスク・スタティック・ストレージのアドレス
*      (a4) =  プロセス・デスクリプタ・アドレス
chkdev:   move.w V_BUSY(a2),d0    デバイスを使用中のプロセス ID
          bne.s isbusy            だれかが使用している
isfree    move.w P$ID(a4),V_BUSY(a2)  デバイスを占有する
isfree10  rts
*
isbusy    cmp.w P$ID(a4),d0       自分のプロセス ID と比較
          beq.s isfree10          自分が使用中
          os9 F$IOQu              I/O キューに入る
          bcs.s isfree10          エラー
          move.w P$Signal(a4),d1  シグナルを受信？
          beq.s chkdev10          No
          cmpi.w #S$Intrpt,d1     致命的なシグナル？
          bls.s chkdeverr         致命的
chkdev10  btst.b #Condemn,P$State(a4) プロセスが殺された？
          beq.s chkdev            殺されていない — 再試行
chkdeverr ori.b #Carry,sr         エラーを通知
          rts
```

〔図19〕SCF の相互排除のデメリットの例

マネージャの設計によっては，別の排他制御が望まれる場合もあります．

たとえば現在の SCF は，あるプロセスがある SCF デバイスからデータを読み出す(I$Read/I$ReadLn)ためにデータの入力を待っている状態で，別のプロセスが同じデバイスに書き込もうとすると，後からリクエストしたプロセス(書込みプロセス)は先に読み出そうとしたプロセス(読出しプロセス)が読出しを完了してファイル・マネージャからカーネルに戻るまで IO キューで待たされることになります．実際，/term と /tl の二つのターミナルを接続した OS-9 システムで，両方のターミナルで TSMon によりログインして，Shell がキーボード入力待ちの状態で /term から

echo hello >/tl

と入力しても，/tl のターミナルでキャリッジ・リターンを押すまで hello のメッセージは表示されず，/term 側では Shell のプロンプトが表示されません．これは，/tl 側の Shell のプロセスが /tl からの入力を待っていて，同じデバイスに出力しようとした echo のプロセスが IO キューに入って待たされているからです(**図19**)．

通信の入力と出力は非同期に発生し，しかも迅速な処理が期待されますから，通信関係の処理を行うファイル・マネージャでは特別の配慮が必要です．VBF (Variable Block File Manager)と MCF(Multi-channel Character File Manager)は，入出力が非同期に動作できるような相互排除を行うことができます．NFM でも当然のことながら，同様の配慮がされているものと考えられます．

CD-RTOS 用の CDFM(Compact Disk File Manager)では，CD のランダム・アクセス速度が遅いことを考慮して，エレベータ・ロジックを採用していることは，Appendix 中(p.151)で述べたとおりです．

また，たんにデバイス・ドライバがある機能をサポートしているかどうかを問い合わせたり，DIP スイッチの設定のような純静的データを読み込むためにデバイス・ドライバを呼び出すときには，相互排除は不要です．RAM ディスクのように「動作の完了を待つ」ことが絶対ない場合も，デバイス・ドライバ内ではコンテクスト・スイッチングは起こりえません．しかし，一般的にはこれらの操作を特別扱いして相互排除を行わないことによる利益はほんのわずかですから，通常は上記の通信操作のような特別の事情がなければ，デバイス・

〔表 2〕デバイス・ドライバの入力レジスタ・パラメータ

(1) INIT，TERM(これらのエントリはファイル・マネージャからは呼び出されない)

レジスタ	用　　途
(a1)	デバイス・デスクリプタのアドレス
(a2)	デバイス・スタティック・ストレージのアドレス
(a6)	システム・グローバル・ベース・アドレス

(2) READ，WRITE

レジスタ	用　　途 代替データ	使用しているファイル・マネージャ
d0.b	データ・バイト —	SCF
d0.1	データ・サイズ(バイトまたはブロック) —	RBF，SBF，IBF
d2.1	データ・ブロック(セクタ)番号 —	RBF
(a0)	データ・バッファのアドレス PD_BUF(a1)	SBF，IBF
(a1)	パス・デスクリプタのアドレス —	全ファイル・マネージャ
(a2)	デバイス・スタティック・ストレージのアドレス V$STAT(PD_DEV(a1))	ほとんどのファイル・マネージャ
(a3)	ドライブ・テーブルのアドレス (a2)+DRVBEG+PD_DRV(a1) ＊DRVMEM	SBF
(a4)	プロセス・デスクリプタのアドレス D_Proc(a6)	ほとんどのファイル・マネージャ
(a5)	ユーザ・レジスタ・スタックのアドレス PD_RGS(a1)	RBF，IBF など
(a6)	システム・グローバル・ベース・アドレス —	全ファイル・マネージャ

(3) GETSTA，PUTSTA

レジスタ	用　　途 代替データ	使用しているファイル・マネージャ
d0.w	SS_＊＊＊機能コード R$dl(a5)	ほとんどのファイル・マネージャ
(a1)	パス・デスクリプタのアドレス —	全ファイル・マネージャ
(a2)	デバイス・スタティック・ストレージのアドレス V$STAT(PD_DEV(a1))	ほとんどのファイル・マネージャ
(a3)	ドライブ・テーブルのアドレス DRVBEG(a2)+PD_DRV(a1) ＊DRVMEM	SBF など
(a4)	プロセス・デスクリプタのアドレス D_Proc(a6)	ほとんどのファイル・マネージャ
(a5)	ユーザ・レジスタ・スタックのアドレス PD_RGS(a1)	RBF，IBF など
(a6)	システム・グローバル・ベース・アドレス —	全ファイル・マネージャ

ドライバを呼び出すときには，つねに F$IOQu による相互排除をしておいたほうがよいでしょう．

4.3　パラメータ

表 2 に，ファイル・マネージャがデバイス・ドライバを呼び出すときのレジスタの設定を示します．パラメータには，すべてのファイル・マネージャで必要なものと，ファイル・マネージャによりサポートする内容が異なるものがあります．また，エントリによってもパラメータは異なります．INIT と TERM のエントリは，カーネルから直接実行され，ファイル・マネージャが実行することはありません．

新しいファイル・マネージャを開発するときは，必ずしも表のレジスタ使用方法にしたがう必要はありませんが，ことさら独自性を発揮する理由もないでしょう．またデバイス・ドライバ内で渡されたレジスタの内容を破壊されてもかまわないように，デバイス・ドライバを呼び出す前に必要なレジスタ(とくにアドレス・レジスタ)の内容はスタックに保存しておきます．

表 2 中の「代替データ」とは，他のレジスタで指されるデータ構造から間接的に得られる値です．表を見てわかるように，アドレス・レジスタのうち必須のレジスタは a1(パス・デスクリプタ)と a6(システム・グローバル・ベース)です．他のアドレス・レジスタの値は，自分以外のアドレス・レジスタから得ることができますから，必ずしもファイル・マネージャが用意しなくてもよいのですが，a2(デバイス・スタティック・ストレージ)と，a4(プロセス・デスクリプタ)は，ほとんどのデバイス・ドライバが使用しますから，ファイル・マネージャ側で用意することが多いようです．

5　ファイル・マネージャの設計

5.1　ファイル・マネージャ開発の必要性

新しいファイル・マネージャの開発は，(少なくとも概念的には)かなり大きなプロジェクトです．「ファイル・マネージャの設計」というタイトルの記事のわりには，はなはだ逆説的ですが，どうしても必要性がなければ避けたほうが懸命です．新ファイル・マネージャ開発の要件としては，おおむね以下のような事項を目安にするとよいでしょう(p.147 コラム 2 参照)．

① 複数の異なるハードウェアに対して論理的な共通

操作の仕様を定義できる.

② 複数プロセスに対して共用資源を提供する必要がある.

③ 複数プロセスが同一資源にアクセスしたときの相互排除処理が必要.

④ I$Read/I$ReadLn/I$Write/I$WritLn のシステム・コールに対応する処理が他のファイル・マネージャと互換性がある.

①は，ファイル・マネージャの性格からしてすぐに思いつくことです．たしかに，共通ライブラリ関数の仕様を決めても，ハードウェアごとにライブラリ関数の内部操作が異なるのでは，異なるハードウェアにプログラムを移植するときには，そのハードウェア用のライブラリをリンクしなおさなければなりません．しかしこの問題は，ファイル・マネージャを開発しなくても，トラップ・ハンドラでも解決することができます．すなわち，呼出しインターフェースは同じだが，ハードウェアに合わせた操作内容のトラップ・ハンドラをハードウェアごとに用意すればよいのです(文献9).

②は，OS-9では各プロセスの使用するスタックやデータ・メモリ，すなわちプログラム・コード以外のメモリは，プロセスごとに独立しており，互いに他のプロセスのデータ・メモリなどをアクセスすることができないことです．この制限は，たんに「行儀」の問題だけでなく，SPU(システム・プロテクション・ユニット)のようなハードウェアのメモリ保護機構をもつシステムでは，自分がアクセス許可をもっている以外の領域は物理的にもアクセスできません(もしアクセスしようとするとバス・エラーが発生する)．複数プロセスから共通にアクセスできるメモリ領域を確保するためには，ファイル・マネージャやデバイス・ドライバなどのシステム・モード・プログラムを経由することも一つの手段ですが，ユーザ・モードのままでも，データ・モジュールを使用して複数プロセス間で共通データをもつことができますから，デバイス・ポートという絶対アドレスを扱うこと以外，②も決定的とはいえません．

③は，②と背理関係にあります．ユーザ・モードでは，割込みを禁止できませんから，いつプロセスが切り替わるか予測できませんので，複数プロセスの相互排除はできません．しかし，イベント・セマフォを使用すれば，ユーザ・モード・プログラムでも共通資源の相互排除が可能ですから，これも決め手にはなりません(文献5).

④は，簡単にいえば

```
$ echo hello >/そのデバイス
```

または

```
$ list /そのデバイス
```

とやって意味があるかどうかということです．これだけは，他の手段では実現不可能で，ファイル・マネージャを導入しなければなりません．

5.2 ファイル・マネージャの設計例

ファイル・マネージャの設計例として，VSFのソース・コードを**リスト3**(稿末)に示します(注)．VSF(Very Simple File Manager)は，OS-9のファイル・マネージャとして最低限の機能をもったファイル・マネージャで，リアルタイム信号処理システムの制御用に開発されました．VSFには以下の特徴があります．

- OS-9のファイル・マネージャとして最低限の機能.
- Open時にパス・リストの解析をデバイス・ドライバでできるように，デバイス・ドライバのSS_OpenのPUTSTAを実行する.
- Read/Writeはユーザ・バッファをそのままデバイス・ドライバに渡し，ブロック転送できる.
- SS_Optなど，通常はファイル・マネージャが行う操作もデバイス・ドライバに渡し，デバイス・ドライバがその機能をサポートしていないのでE$UnkSvcを返してもエラーを無視する.

VSFの仕様は，マイクロウェア・ジャパンから発売されているSBBF(Sequential Binary Block File Manager)とほぼ同じです．じつは，当初はSBBFの採用も検討したのですが，SBBFの当時のバージョンにはいくつかの問題(パスのオープン時にデバイス・ドライバが実行するSS_OpenのPUTSTAが返すエラーをすべて無視するという問題など)があったため，新たにファイル・マネージャを開発することになったという経緯があります．

(注) 読者諸兄がVSFを使用するにあたっては，個人や学校，企業などにおける実験や単独使用には自由に使用してかまいませんが，同一事業所の複数台のコンピュータで使用したり，第三者に対してVSFを含むソフトウェアを販売・譲渡する場合には，㈱エー・アール・ケー・コーポレーションとライセンス契約(有償)を結ぶ必要がありますのでご注意ください．

5.3 OS-9とC言語

つい最近までは，ファイル・マネージャやデバイス・ドライバのような「システム・レベル」のプログラムは，アセンブリ言語で書かれるのが当然でしたが，最近は高級言語(とくにC言語)で書かれることが多くなりました．これには，① Unixの成功，② コンパイラ技術の進歩，③ ハードウェア(CPU)の高速化，④ ソフトウェアとハードウェアの提供者の分離(移植性)，⑤ 開発費の高騰を抑える(プログラマ不足)，⑥ 要求の肥大化，などのような理由が考えられます．

OS-9もこの例にもれず，「アセンブリ言語で書かれたコンパクトなOS」をキャッチフレーズにしてきた時代とは，多少ようすが違ってきました．たとえば，OS-9/ISPのファイル・マネージャはC言語で書かれています．今年中にもリリースが予定されているMS-DOSディスク用ファイル・マネージャMSFMも，Cで書かれています(文献8)．マイクロウェアのKim Kempf氏によれば，fprintf()などのライブラリ関数を，ファイル・マネージャなどのシステム・プログラム用に，スタックをあまり使用しないように書き直したそうです．サード・パーティの製品ではありますが，Windsor社のMFM(Memory File Manager)もCで書かれています．OS-9 ver 2.2の開発用ライセンスに含まれるC言語用の定義ファイルには，「Cで書かれたファイル・マネージャ用」の定義が含まれています．

さらに，手元に届いたばかりの最新版のカーネル(ver 2.3)は，コード・サイズがver 2.2の18Kバイトから25Kバイトへと大幅に増加しただけではなく，明らかにCコンパイラの生成したコード・パターンが含まれています．これらの動きは，噂されているOS-9のMC680x0以外のMPUへの移植とは無縁ではないでしょう．今回開発したVSFは，処理内容がほとんど何もないに等しいので，アセンブリ言語で書き，あえてCでは書きませんでしたが，じつはVSFによって駆動されるデバイス・ドライバはCで書かれています．もし読者諸兄の中に新しいファイル・マネージャを開発してみようとお考えの向きがあれば，できるだけC言語で書くことをお勧めします．

むすび

今回の執筆にあたり，いくつかのファイル・マネージャを解析しなおしてみました．そして，改めてOS-9の構造の美しさに感動し，Robert Dogget氏をはじめとする設計者たちに対する尊敬の念をいっそう深めることになりました．とくにRBFなどは，それ自体が複数プロセスからの資源要求を調停する小さなOSともいえます．今回すべてを紹介できなかった各ファイル・マネージャの動作の詳細はもとより，システム・コールの実現方法，メモリ管理，プロセス管理，モジュール管理，割込み管理など，研究に値する題材はいくらでもあります．ソース・コードに触れるチャンスがなければ，なかなか難しいかもしれませんが，その気になればデバッガによるメモリ・ダンプと逆アセンブルでもかなりのことが解析できます．

OS-9は，手頃でかつ本格的なOSとして，とにかく興味の尽きないOSです．噂によれば，MC680x0ファミリ以外のMPUへの移植作業も進行中とのことですし，今後も「正統派」のOSの一つとして，独自の世界を広げていくことでしょう．

参考文献

1) 箕原辰夫,『OS-9/6909 I/O解析マニュアル』, 秀和システム・トレーディング, 1985年3月
2) *CD-I Full Functional Specification*, N.V.Philips, Sony Corporation, March 1987
3) *OS-9/68000 Technical Manual Revison H*, Microware Systems Corporation, June, 1987
4) 菅原宏和:「CD-RTOSの概要」,『OS-9 NEWS』, No.17, ㈱星光電子, 1988年3月
5) 菅原宏和,「OS-9/68000イベント機能の使い方」,『OS-9 NEWS』, No.18, ㈱星光電子, 1988年6月
6) 菅原宏和,「IBF-IEEE488/GP-IBファイル・マネージャ」,『パイプラインズ・ジャパン』, No.1, マイクロウェア・ジャパン㈱, 1988年9月
7) 菅原宏和,「OS-9 ⟷ UNIXの通信」,『OS-9 NEWS』, No. 20, ㈱星光電子, 1988年12月
8) Dibble, Peter, *OS-9 Insights*, Microware Systems Corporation, 1988
9) 菅原宏和,「TRAPハンドラ」,『パイプラインズ・ジャパン』, No.2/3, マイクロウェア・ジャパン㈱, 1988年12月, 1989年3月
10) 菅原宏和,「システムステート・プロセスのパス番号」,『OS-9 NEWS』, No.21, ㈱星光電子, 1989年3月
11) 菅原宏和,『IEEE488/GP-IB設計マニュアル』, ㈱総合電子出版, 1989年7月

すがわら・ひろかず ㈱エー・アール・ケー・コーポレーション

〔リスト3〕ファイル・マネージャの例(VSF) ①

① 定義部

```
        ttl     VSF             File Manager
        nam     Definitions
****************************************************************
*
*   VSF - Very Simple File Manager for OS-9/68000
*       Copyright 1989 Hiro Sugawara/ARK Corporation
*
*   Module: vsf_a - Definitions for library use
*
*  #  Reason for change                          By   Date
* ---- ------------------------------------------ ---- --------
*  1  Modified  from IBF ed#10                   hiro 5/12/89
*
Edition   equ     1               Current edition number

        psect   vsf_a,0,0,Edition,0,0
        use     defsfile
        opt     -l

*-----------------------------------------
* Offsets for Kernel use
*
PD_FST    equ     $2a

*-----------------------------------------
* Path Descritor Definitions
*
IE_Max:   equ     16        max event slots
IE_Last:  equ     IE_Max-1  last event code

        org     PD_FST    File manager's area
PD_MAX:   do.w    1         maximum number of bytes
                                    for ReadIn (unused)
PD_RAW:   do.b    1         flag to distinguish
                                  Read(!=0)/ReadIn(=0)
          do.b    $40-.     Reserved
PD_EvSig: do.w    IE_Max    event signal table
PD_EvMask: do.w   1         event registration mask
          do.b    128-.     Reserved
PD_DTP:   do.b    1         device class (= 25)
          do.b    1
PD_REOS:  do.b    1         end-of-sequence character on read

*-----------------------------------------
* GetStat/SetStat function codes
*
          org     416
SS_VSSig: do.b    1             control signals on events
          do.b    431-.         reserved

*-----------------------------------------
* miscelleneous constants
*
DT_VSF:   equ     25            VSF device type code

          ends
```

② Main ルーチン

```
        ttl     VSF             File Manager
        nam     Main            Entry
****************************************************************
*
*   VSF - Very Simple File Manager for OS-9/68000
*       Copyright 1989 Hiro Sugawara/ARK Corporation
*
*   Module: ibfmain_a - Main Entry
*
*  #  Reason for change                         By   Date
* ---- ----------------------------------------- ---- --------
*  1  Modified from IBF ed#27                   hiro 5/12/89
*

Edition   equ     1               current edition number
Revision  equ     1               current revision number

          use     defsfile
          opt     -l

TypLan    equ     (FlMgr<<8)+Objct
AttRev    equ     ((ReEnt+SupStat)<<8)+Revision

          psect   vsfmain_a,TypLan,AttRev,Edition,0,Entry

          dc.b    "Copyright 1989 ARK Corporation",0

*-------------------------------------------------------
* Entry Offset Table (entries from Kernel)
*    input
*       (a1)  ptr to path descriptor
*       (a4)  ptr to current process descriptor
*       (a5)  ptr to user's register stack
*       (a6)  ptr to system global area
*
Entry     dc.w    Create-Entry
          dc.w    Open-Entry
          dc.w    MakDir-Entry
          dc.w    ChgDir-Entry
          dc.w    Delete-Entry
          dc.w    Seek-Entry
          dc.w    Read-Entry
          dc.w    Write-Entry
          dc.w    ReadIn-Entry
          dc.w    WriteIn-Entry
          dc.w    GetStat-Entry
          dc.w    PutStat-Entry
          dc.w    Close-Entry
*
* Unsupported entries
*
MakDir:   equ     *
ChgDir:   equ     *
Delete:   equ     *
ErrUnkSvc: move.w #E$UnkSvc,dl
ErrRet:   ori.w   #Carry,sr
          rts
*
Seek:     moveq.l #0,dl
          rts


          ttl     Common          miscellaneous routines

*-------------------------------------------------------
* CallDriver - Common for calling dirver's entry
*    input
*       d0.l  message buffer length (for Read & Write)
*       d0.w  function code (for GetStat & PutStat)
*       d5    *** DO NOT USE TO PASS PARAMETER ***
*       d7.l  D$READ,D$WRIT,D$GSTA,D$PSTA
*       (a0)  function dependent
*       (a1)  ptr to path descriptor
*       (a2)  ptr to static storage area
*       (a4)  ptr to current process descriptor
*       (a5)  ptr to user's register stack
*       (a6)  ptr to system global area
*    output
*       cc    set if an error
*       dl.w  Error code
*    destroyed
*       d0-d7
*
CallDriver: move.l d0,d5          ChkBusy destroys
CDriver10 move.w  V_BUSY(a2),d0
          beq.s   CDriver30       device is free
          cmp.w   P$ID(a4),d0     am I using ?
          beq.s   CDriver30       yes
          os9     F$IOQu          device is busy,
                                        enter I/O queue
          bcs.s   CDriver25       some error, return quickly
```

```
         move.w   P$Signal(a4),d1 get signal
         cmpi.w   #S$Wake,d1      wake up signal ?
         bls.s    CDriver20       continue if so
         cmpi.w   #S$Intrpt,d1    deadly signal ?
         bls.s    CDriverErr      ..Yes; give up
CDriver20 btst.b  #Condemn,P$State(a4) has process died ?
         beq.s    CDriver10       check device again
CDriverErr ori.w  #Carry,sr
CDriver25 rts
*
CDriver30 move.l  d5,d0           recover
         movem.l  a0-a6,-(sp)     save registers
         move.w   P$ID(a4),V_BUSY(a2) occupy device
         move.w   P$ID(a4),V_LPRC(a2)
         movea.l  PD_DEV(a1),a3   get device table entry
         movea.l  V$DRIV(a3),a3   get device driver
                                               module address
         add.l    M$Exec(a3),d7   offset to offset table entry
         move.w   (a3,d7.w),d7    entry offset
         jsr      (a3,d7.w)       call the entry
         movem.l  (sp)+,a0-a6     recover registers
         move.w   sr,-(sp)        save carry
         clr.w    V_BUSY(a2)      free device
         rtr

         ends
```

③ Open/Create/Close ルーチン

```
         ttl      VSF             File Manager
         nam      Open/Create/Close Routines

* Edition records kept in vsfmain.a

         psect    vsfopen_a,0,0,0,0,0
         use      defsfile
         opt      -l

*-------------------------------------------------------
* Open   :  Open device path   - I$Open
* Create :  Create device path - I$Create
*    input
*        R$d0(a5).b  access mode
*        R$a0(a5) ptr to pathlist
*        (a1) ptr to path descriptor
*        (a4) ptr to current process descriptor
*        (a5) ptr to user's register stack
*        (a6) ptr to system global area
*    error
*        cc   set
*        d1.w error code
*
Open:
Create:
*
* parse pssed pathlist
*
         movea.l  R$a0(a5),a0     Get pathlist pointer
         move.l   a1,-(sp)        Save path descriptor pointer
Open02   os9      F$PrsNam
         bcs.s    Open05
         eori.b   #PDelim,d0      continue ?
         bne.s    Open05          no (carry always cleared)
         movea.l  a1,a0           parse next element
         bra.s    Open02
*
Open05   movea.l  a1,a3           updated pathlist
         movea.l  (sp)+,a1
         bcs.s    Open60          error
*
* check pathlist terminator
*
Open10   move.b   (a3),d0         get the delimiter
         beq.s    Open20          null character
         cmpi.b   #C$Spac,d0      Space character ?
         beq.s    Open20
         cmpi.b   #C$CR,d0        carriage return ?
         beq.s    Open20
         cmpi.b   #C$Tab,d0       tab ?
         beq.s    Open20
         move.w   #E$BPNam,d1
         bra.s    Open56          signal error

*
* clear all signal table entries
*
Open20
         lea.l    PD_EvSig(a1),a0
         moveq.l  #IE_Max+1,d1    include PD_EvMask
         bra.s    Open50
*
Open40   clr.w    (a0)+
Open50   dbra     d1,Open40
*
* call device driver to inform opening
*
         movem.l  a3/a5,-(sp)     save updated pathlist
         movea.l  PD_DEV(a1),a3
         movea.l  V$STAT(a3),a2
         move.w   #SS_Open,d0     opening call
         bsr      PS_Driver
         movem.l  (sp)+,a3/a5     recover updated pathlist ptr
         move.w   sr,d7           save carry
         move.l   a3,R$a0(a5)
         move.w   d7,sr
Open55   bcc.s    Open60          no error
         cmpi.w   #E$UnkSvc,d1    does driver support SS_Open ?

         beq.s    Open60          ignore it
Open56   ori.w    #Carry,sr       signal error
Open60   rts


         ttl      Close           path

*-------------------------------------------------------
* Close  :  Close device path   - I$Close
*    input
*        (a1) ptr to path descriptor
*        (a4) ptr to current process descriptor
*        (a5) ptr to user's register stack
*        (a6) ptr to system global area
*    error
*        cc   set
*        d1.w error code
*
Close:
         tst.b    PD_CNT(a1)      last image ?
         beq.s    Close02         yes
         rts                      do nothing

Close02  movea.l  PD_DEV(a1),a3   get device table entry
         movea.l  V$STAT(a3),a2   get device
                                      static storage address
         move.w   #SS_Close,d0    call driver entry
         bsr      PS_Driver
         move.w   sr,-(sp)        save error status
         move.w   P$ID(a4),d0
         cmp.w    V_LPRC(a2),d0   am I the last user ?
         bne.s    Close20         no
         clr.w    V_LPRC(a2)      clear last user
Close20  move.w   (sp)+,sr
         bra.s    Open55

         ends
```

〔リスト3〕ファイル・マネージャの例(VSF) ②

④ Read/Readln/Write/Writeln ルーチン

```
         ttl      VSF           File Manager
         nam      Read/Readln/Write/Writeln

* Edition records kept in vsfmain.a

         psect    vsfread_a,0,0,0,0,0
         use      defsfile
         opt      -l

*-----------------------------------------------------------
* Read:     Read raw bytes  - I$Read
* Readln:   Read text bytes - I$Readln
*   input
*        (a1)       ptr to path descriptor
*        (a4)       ptr to current process descriptor
*        (a5)       ptr to user's register stack
*        (a6)       ptr to system global area
*        R$d1(a5).l  max number of bytes to read
*        R$a0(a5)    ptr to user's buffer
*   output
*        R$d1(a5).l  number of bytes actually read
*   error
*        cc          set
*        d1.w        error code
*
Readln:  clr.b    PD_RAW(a1)
         bra.s    ReadVSF
*
Read:    move.b   #YES,PD_RAW(a1)
ReadVSF  move.l   R$d1(a5),d1    Max number of bytes to read
         beq.s    ReadRet        No bytes, return quickly
Read12   movea.l  PD_DEV(a1),a3  Get device table address
         movea.l  V$STAT(a3),a2  Get static storage address
Read30   move.l   R$d1(a5),d0    Max number of bytes to read
         move.l   R$a0(a5),a0    User's buffer address
         moveq.l  #D$READ,d7     Say call driver's read entry
         bsr      CallDriver     Call driver
         bcs.s    ReadRet
*
* I$Readln process
*
         tst.b    PD_RAW(a1)     I$Read ?
         bne.s    Read50         Yes
         tst.l    d0             How many bytes read ?
         beq.s    Read50         None
         cmpi.b   #C$CR,PD_REOS(a1) C/R expected to terminate ?
         beq.s    Read50         Yes, no problem
         move.b   PD_REOS(a1),d1 Get the terminator
         cmp.b    -1(a0,d0.l),d1 Did it actually
                                   terminate the transfer ?
         bne.s    Read50         No, max transfer size
                                                terminated
         cmpi.b   #1,d0          Only the character ?
         beq.s    Read40         Yes, terminator character only
         cmpi.b   #C$CR,-2(a0,d0) C/R,L/F type termination ?
         bne.s    Read40         No (maybe L/F only)
         subi     #1,d0          C/R,L/F type,
                                      ignore last character
         bra.s    Read50
*
Read40   move.b   #C$CR,-1(a0,d0) Change the terminator to C/R
Read50   move.l   d0,R$d1(a5)    Copy actual number
                                             of bytes read
ReadRet  rts


         ttl      Write/Writeln
*-----------------------------------------------------------
* Write:    Write raw bytes
* Writeln:  Write text bytes - I$WritLn
*   input
*        (a1)       ptr to path descriptor
*        (a4)       ptr to current process descriptor
*        (a5)       ptr to user's register stack
*        (a6)       ptr to system global area
*        R$d1(a5)   max number of bytes to write
*        R$a0(a5)   ptr to user's buffer
*   output
*        R$d1(a5)   number of bytes actually written
*   error
*        cc         set
*        d1.w       error code
*
Writeln:  clr.b   PD_RAW(a1)
          movea.l R$a0(a5),a0    get source ptr
          moveq.l #C$CR,d2
          move.l  R$d1(a5),d0    get maximum
          bra.s   Writeln20      scan for C$CR

Writeln10 cmp.b   (a0)+,d2       carriage return ?
Writeln20 dbeq    d0,Writeln10   no
          beq.s   Writeln30      continue if cr found
          addq.w  #1,d0          propagate carry
          subq.l  #1,d0
          bcs.s   WriteVSF       maximum reached (no cr)
          bra.s   Writeln10      check next 64k bytes

Writeln30 neg.l   d0             get negated bytes
                                          not transfered
          add.l   R$d1(a5),d0    up to cr
          bra.s   WriteVSF
*
Write:    move.b  #YES,PD_RAW(a1)
          move.l  R$d1(a5),d0    how many bytes to write ?
WriteVSF  tst.l   d0
          beq.s   ReadRet        No bytes, return quickly
          move.l  R$a0(a5),a0    User's buffer address
          movea.l PD_DEV(a1),a3  Get device table address
          movea.l V$STAT(a3),a2  Get static storage address
          moveq.l #D$WRIT,d7     Say call driver's write entry

          bsr     CallDriver     Call driver
          bcs.s   WriteRet
          move.l  d0,R$d1(a5)    Copy actual number
                                          of bytes written
WriteRet  rts

          ends
```

⑤ SetStat/GetStat ルーチン

```
          ttl     VSF            File Manager
          nam     Set/Get        Status Routines

* Edition records kept in vsfmain.a

          psect   vsfsgst_a,0,0,0,0,0
          use     defsfile
          opt     -l

          ttl     Put            Status
*-----------------------------------------------------------
* PutStat:  Put device status - I$SetStt
*
* Supported:  SS_Opt,SS_VSSig
*   input
*        (a1)      ptr to path descriptor
*        (a4)      ptr to current process descriptor
*        (a5)      ptr to user's register stack
*        (a6)      ptr to system global area
*        R$d1(a5)  SS_xxx function code
*        R$**(a5)  function dependent
```

```
*    output
*         R$**(a5)  function dependent
*    error
*         cc        set
*         dl.w      error code
PutStat:   movea.l  PD_DEV(a1),a3  Get device table address
           movea.l  V$STAT(a3),a2  Get static storage address
           move.w   R$d1+2(a5),d0  See the function code
*
* Entry for internal calls
*    input
*         d0     SS_*** function code
*         (a5)   register stack for parameters (if any)
*
PutSta:
           cmpi.w   #SS_Opt,d0     set option area ?
           beq.s    PS_Opt
           cmpi.w   #SS_VSSig,d0
           beq.s    PS_VSSig
*
* Additional PutStat functions processed here
* -- No more VSF supported entries, call driver
PSta10     moveq.l  #D$PSTA,d7     Offset for PutSta entry
           bra      CallDriver     Call driver

* -- Common internal driver calling entry
*    input: d0.w  SS_*** code
PS_Driver: move.l   R$d1(a5),-(sp) save user's register
           move.w   d0,R$d1+2(a5)  set SS_*** code
           bsr.s    PSta10
           move.w   sr,d7          save carry
           move.l   (sp)+,R$d1(a5) recover original user register
           move.w   d7,sr
           rts


           ttl      SS_VSSig       - signal registration

*-------------------------------------------------------
* SS_VSSig - Set signals according to events
*    input
*         R$d2+2(a5).w  Signal code
*         R$d3+2(a5).w  Event code (0..15)
*
PS_VSSig:  move.w   R$d3+2(a5),d3
           cmpi.w   #IE_Last,d3
           bhi      ErrUnkSvc      too large !
           bsr.s    PS_Driver      ask if valid code
           bcs.s    PS_VSSigRet
           move.w   PD_EvMask(a1),d4 get current mask
           move.w   R$d2+2(a5),d2  get signal code
           beq.s    PS_VSSig15
           bset.l   d3,d4          set PD mask
           move.w   V_EvMask(a2),d5 get master mask
           bset.l   d3,d5          set mask
           move.w   d5,V_EvMask(a2) restore it
           bra.s    PS_VSSig20
*
PS_VSSig15 bclr.b   d3,PD_EvMask(a1) set mask
PS_VSSig20 move.w   d4,PD_EvMask(a1) restore mask
           asl.w    #1,d3          shift left for word indexing
           move.w   d2,PD_EvSig(a1,d3.w) set it into signal table
PS_VSSigRet rts

*-------------------------------------------------------
* SS_Opt - Set path descriptor's option area
*         R$a0(a5)  ptr to user's option area buffer
*
PS_Opt:    movea.l  R$a0(a5),a0    get user's buffer address
           bsr.s    PS_Driver
           bcc.s    PS_OptRet
           cmpi.w   #E$UnkSvc,d1   does driver support ?
           bne      ErrRet
PS_OptRet  rts                     ignore SS_Opt
```

```
           ttl      Get            Status
*---------------------------------------------------------
* GetStat:  Get device status - I$GetStt
*           Supported functions: SS_Opt
*    input
*         (a1)      ptr to path descriptor
*         (a4)      ptr to current process descriptor
*         (a5)      ptr to user's register stack
*         (a6)      ptr to system global area
*         R$d1(a5)  SS_xxx function code
*         R$**(a5)  function dependent
*    output
*         R$**(a5)  function dependent
*    destroyed
*         d0-d7/a2-a3
*    error
*         cc        set
*         dl.w      error code
*
GetStat:   movea.l  PD_DEV(a1),a3  Get device table address
           movea.l  V$STAT(a3),a2  Get static storage address
           movem.l  R$d1(a5),d0    See the function code
*
GetSta:
           cmpi.w   #SS_Opt,d0     peeping path descriptor
                                                 option area ?
           beq.s    GS_Opt
*
* Additional SS_*** codes processed here
* -- No more entries
GSta10     moveq.l  #D$GSTA,d7     Offset for GetSta entry offset
           bra      CallDriver     VSF does not support
                                        this function, call driver

* -- Common internal driver calling entry
*    input: d0.w  SS_*** code
GS_Driver: move.l   R$d1(a5),-(sp)
           move.w   d0,R$d1+2(a5)  set SS_*** code
           bsr.s    GSta10
           move.w   sr,d7          save carry
           move.l   (sp)+,R$d1(a5) recover original user register
           move.w   d7,sr
           rts

*--------------------------------------------------------
* SS_Opt - return path descriptor option area
*
GS_Opt:    movea.l  R$a0(a5),a0    get user's buffer address
           bsr.s    GS_Driver
           bcc.s    GS_OptRet
           cmpi.w   #E$UnkSvc,d1   does driver support ?
           bne      ErrRet         real error
           movea.l  a0,a2          remember destination pointer
           lea.l    PD_OPT(a1),a0  beginning of option area
           move.l   #128,d2        size of the area
           os9      F$Move         copy bytes
GS_OptRet  rts

           ends
```

X68000への OS-9/68Kの移植

菅野　聡

すでにX68000用にはOS-9/X68000が市販されているが，ここではOS-9/68KをX68000に移植した事例を紹介する．移植は，基本的には先月紹介したROMベース・システムを構築する場合と同様であって，ブートストラップ・プログラムと物理的にI/Oを制御するためのデバイス・ドライバを作成することがおもな作業内容である．ただしディスク・ベース・システムの場合には，まずブートROMからディスク上のIPLをロードし，そのIPLからブートストラップ・プログラムを実行するようにする．またデバイス・ドライバでは，レスポンスの速いマルチタスキングを実現するために，I/O待ちが起こるような場合には，デバイスの特性を考慮しながらsleepをかける必要がある．なお今回の移植は，Level 1 ver 1.2のOS-9/68Kを使用したが，デバイス・ドライバなどは，それ以降のバージョンのものと互換性がある．　（編集部）

本稿ではFM-11(OS-9カード)をホストとし，シャープのX68000をターゲットとしたOS-9/68000 Level1バージョン1.2の移植事例を紹介し，実際のソース・プログラムについて解説します．

1 プロローグ

クリスマス当日，下界の賑わいから切り離され人の気配が消え去った静かな5階の一室．オレンジ色に染まった部屋の天井を仰ぎ，ため息をつきながら冷めたコーヒーをすする．いつ終わるかしれないホスト，ターゲット間のシャトル・ディスケット号．デバッグの見当も付かぬまま散乱したディスケットの中から選ばれた1枚がホストにセットされる．scredをexし，

make OS9Boot ↵

合体ロボットFM-11(OS-9/68K)が目をひからせうなり始める．順にトラックを滑るヘッドの音を聞きながら，リスト用紙のカーペットを眺め「あーあああああ」(訳：もういやだ)と口をつく．セクタ・ライトを終えたディスケット号がターゲットに吸い込まれ長いディスク・リード・コマンドをタイプする．

R 4000 9070 0101000E 10000 ↵　　(1)

せめて一回限りのヒストリくらいあればいいのに．マンハッタン・ビルディングの明滅するネオンを見ながらもはや起動への期待に疲れ，指で覚えたコマンドをブラインド・タッチ．今度のエラー・メッセージは何かなとぼんやり端末98号に目をやり，慰めを置いて去った友人たちの言葉を思い出した…．デバッグで年を越すなんて，なんて哀れなんだろう．大きなため息とともに気を取り直して目の焦点をむりやり合わせる．

```
shell version 1.2
$
```

これは一体，何のエラー・メッセージだろう？

▶経緯

現在，シャープからX68000用のOS-9/X68000がリリースされ，その開発用アプリケーションであるCコンパイラもマイクロウェアからリリースされ基本的な環境としてはだいぶ良くなったといえるでしょう．

移植当時，X68000が発売されたばかりで，パソコン・レベルの68Kマシンとしては先陣を切った形となりました．業界の主流といえるインテル，IBM，マイクロソフトという構図の中で，パソコン・レベルの68KマシンはやはりMS-DOSを意識したOSが搭載されていました．しかし，6809から68030へとつづくOS-9の魅力はぬぐい去ることはできません．

好運にもマイクロウェア，フォークスの協力と『マイコンピュータ』(No.25)「OS-9/68000の研究」を執筆された阿部英志氏の支援という環境に恵まれ，山川直巳氏とともに移植の醍醐味を経験することができました．

移植の目標は，もちろんターゲット単体でOS-9が立ち上がること．ターゲット単体でさまざまなアプリケーションを開発できることです．つまり，ターゲットのFDドライバの作成によりディスク・ベースのファイル・システムを構築し，ターゲットのディスプレイ，キーボードを用いたソース・ファイルの編集のため，scredのtermsetファイル内に記述されたターミナル機能(図1)を実現できるターミナル・ドライバを作成することです．

以下では，そのときの経験に基づいて移植のポイントとデバイス・ドライバなどのソース・プログラムについて解説していきます．

2 移植環境

移植に際して，用意できた環境はつぎのとおりで，これを図2に示します．

まず，OS-9/68KのホストはFM-11AD2(フォークスのOS-9/68Kカード)です．このマシンは，制御などの応用分野でOS-9が走るということから，今だに根強く残っています．つい最近まで生産されていたという噂を聞くほどです．

いうまでもなくOS-9の移植はOS-9自身が走るホストがないとほとんど無理です．CP/Mのように BIOSだけを無造作に作り，IPLを作って移植するのとはわけが違います．

OS-9の移植は，CP/Mなどと同じ指向で，ターゲット特有のI/Oを制御するいくつかのプログラムを作りますが，それ自体がOS-9の特徴であり，利点であるモジュール構造を形成していなければなりません．したがって，OS-9の定義ファイルを熟知している者でないかぎり(そんな奴はホストなしで移植なんかを行うはずがない)そんな大それたことはできないでしょう．

〔図1〕スクリーン・エディタscredのスペシャル・エスケープ・シーケンス

- カーソル移動（$1b=¥Y¥X）
- 1行削除（$1bM）
- 1文字削除（$1bP）
- 画面クリア（$1bJ）
- カーソルから画面最下行までクリア（$1bK）
- ライン・インサート（$1bL）
- 文字反転（$1bm）
- 文字反転解除（$1bn）

▶デバッグ環境

ターゲットX68000上で移植に必要なモニタ作業は，ROMで標準装備されたROM Debugger ver 1.0を用いました．

このデバッガは非常に便利で，今回の移植媒体である2HDのFDではさまざまなIBMフォーマットにコマンド・ラインから対応できるため，手作業でも難なく移植できました．また，RS-232-Cを介して端末から操作すると，ベクタやシステム・ワーク・エリアを除いたRAMのほとんどがリセット直前(デバッガ起動)のままで参照できるので，大変重宝しました．

しかし，その本来のデバッグ機能はほとんど使用できませんでした．つまり，OSの移植というスーパバイザでの作業は，システム・スタック・ポインタやベクタをテスト・ランで変更してしまうためブレーク・ポイントやトレース機能は当然利用することができないということです．

また，この端末にはPC-9801のMS-DOS上でC-TERMを用いました．

本来，移植作業効率を考えるとディスク・メディアでの手作業は非常に労力を要するものであり，ターゲットのデバッガがシリアル回線で運用できるのであれば，そのデバッガ用のモニタをホストのOS-9(マルチタスク！)で管理したほうがより効率的であり，スマートであるといえるでしょう．

今回の移植作業の環境は以上のとおりですが，これに加え，ICEがあれば完璧といえます．

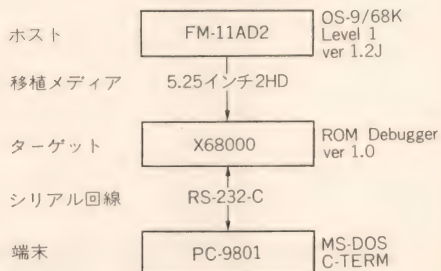
▶X68000のシステム

ここで，X68000の豊富なデバイスのうち移植に関係してくるデバイスについて簡単に解説します(pp.204-206のAppendixも参照)．

● DMAC(HD63450：DMAコントローラ)

FDC用にチャネル0のみを使用しましたが，他に三つもチャネルがあるので，RAMディスクやvtermのテキストRAMアクセスに用いると効果的でしょう．

● MFP(MC68901：多機能ペリフェラル)

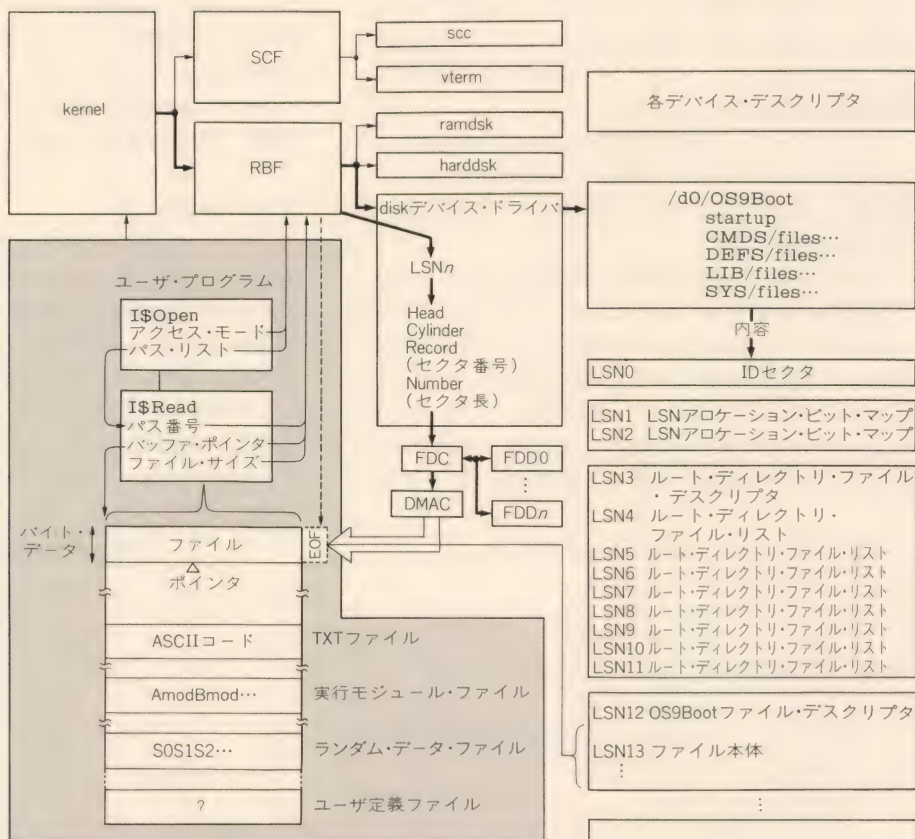
〔図2〕移植環境概略

割込みやタイマ，シリアル通信などの機能を備える汎用の多機能周辺 LSI です．割込みを制御するには，この LSI を操作します．

タイマには，A から D の四つがあります．タイマ A はタイマの中でも長い時間に利用されるもので，今回は FDD のモータ OFF 遅延制御に用いました．タイマ B は MFP 内の USART のクロックに接続されていて，キーボードやマウスを使用するかぎり，他の用途には使えません．タイマ D は OS-9 の鼓動ともいえるタスク切替えに用いました．USART はキーボードとのインターフェース専用です．

● SCC(Z8530：シリアル通信コントローラ)

X68K の ROM Debugger と簡単なプログラムによる各モジュールのテストのために用いました．このデバイスは非常に高機能で各種のシリアル通信に対応しています．しかし，そのぶん設定が複雑です．

● RTC(RP5C15：リアルタイム・クロック)

実時間を得るために用いました．読出しデータはすべて 1 桁の BCD コードです．OS-9 の F$STime では，入力を d0.l(00hhmmss), d1.l(yyyymmdd) レジスタの各バイトごとに 16 進で設定するため，変換しなければなりません．

● FDC(μPD72065：FD コントローラ)

この FDC はマルチシーク，マルチセクタ・アクセス，マルチトラック・アクセスをサポートしています．しかし，今回の OS-9 ver 1.2 では，単一のセクタ・アクセスしか必要ないため，利用しませんでした．ver 2.0 以降ではマルチセクタ・アクセスがサポートされているため，これを利用することができるでしょう．また，この FDC の TC(ターミナル・カウント)端子への入力が重要です．X68K のハード構成ではシステム・コントローラの BUDDHA(カスタム・デバイス)と DMAC からの信号を AND して TC に接続しているため，DMA なしでは FD は利用できません．

● CRTC(カスタム：CRT コントローラ)

このカスタム・デバイスはさまざまな機能を備えていますが，その中のラスタ・コピー機能だけをスクロールに利用しました．

▶移植媒体

はじめに，移植媒体であるディスケットについて説明します．

通常ディスクのファイルは何度も書き換えられて，それによってバラバラに分割され格納されています．

〔図 3〕
ファイル・システム概略

しかし，これでは今回のように移植などの理由から物理的に直接データを読み出す場合には大変です．したがって，今回は一番楽な方法として，ディスクにはターゲットに渡すデータを一つだけ格納し，データを分割させずに書き込ませるという方法を取りました．これはフォーマットしたての，あるいはすべてのディレクトリを含むファイルを消去したディスケットのルート・ディレクトリにファイルをセーブすることにより，そのファイルはシリンダ(0～153)＝1，ヘッド(0，1)＝0，セクタ(1～26)＝14から連続して格納されます．

ターゲットでは，冒頭の(1)で示したディスク・リード・コマンドを入力すれば，$4000から$10000バイトのデータを読み込むことができます．

▶ファイル・システム

ここでOS-9のディスク管理について触れておきます．OS-9のファイル・システムはUnixから受け継いだユニファイドI/Oを用いています．これはさまざまな入出力デバイスをもその論理的機能を定義したファイルとみなし，そのすべてにパス・ネームを割り当て，プログラムの実行時にその入出力をさまざまな形でリダイレクト可能にするものです．ここではとくにディスク・デバイス上のファイルを読み込む場合を解説します．

OS-9では標準のセクタ長を256($100)バイト単位としています．つまり，IBM形式MFMモードでは1トラック当たり26($1A)セクタです．

これらのセクタを順番に論理セクタ番号(LSN0～)で管理しています．この管理はRBFが行い，ディスク上のファイルはすべてこのLSNごとに管理されています．

ユーザが作成したプログラムまたはデータは，あつかう入出力装置の性質をいっさい気にすることなく，一つのファイルとして与えられた(あるいは与える)パスに対して読み書きをするだけです．

RBFはLSNを用いて，ユーザから与えられたパスにしたがってディレクトリを検索し目的とするファイルを管理します．ディレクトリすらファイルという形態をもち，すべてのファイルはその属性，生成日時，オーナIDやファイル・アロケーション情報が格納されたファイル・デスクリプタをもっています．

RBFから受け取った論理セクタ番号から実際のシリンダ，ヘッド，クラスタ(セクタ)といういわば物理セクタ番号への変換は，ディスク・ドライバに一任されています．したがって，ディスク・ドライバはディスク上の物理セクタを読み書きするだけです．

システムによっては，ディスクの最外周(トラック0，1)をシステムのIPL専用に確保してあり，LSNでは管理されません．X68000はこの方式を取ります．

このファイル・システムを簡単に図3に示します．

3 移植の手順

移植作業はつぎの三つの山から成り立っています．

(1) ドライバなどのモジュールの作成
(2) ターゲット上でのテスト
(3) システム・ディスク作成

ターゲットのために作成する新たなモジュールを図4に示します．これらのモジュールを順にホスト上

〔図4〕OS9Bootファイルの内容(オフセット$4000)

```
Addr  Size Type Module name
---- ----- ---- -----------
4000 13076 Sys  kernel
7314   212 Sys  clock
73e8  1846 Fman pipeman
7b1e   102 Desc pipe
7b84   118 Desc nil
7bfa   102 Driv null
7c60  5540 Fman rbf
9204  2262 Driv disk
9ada   114 Desc d0
9b4c   114 Desc d1
9bbe   312 Driv ramdsk
9cf6   112 Desc r0
9d66   112 Desc r1
9dd6  1562 Fman scf
a3f0  3084 Driv vterm
affc   120 Desc term
b074   218 Sys  init
b14e   220 Prog sysgo
```

〔図5〕システム・ディスクの内容

```
/d0/……ルート・ディレクトリ
 OS9Boot……ブート・ファイル
 startup……プロシージャ・ファイル

 CMDS/……実行モジュール・ディレクトリ
  cio  dd.d0  dd.r0  dd.r1  iniz  shell...

 DEFS/……定義ファイル・ディレクトリ
  ctype.h  defsfile  dir.h  direct.h  errno.h
  math.h  modes.h  module.h  oskdefs.d
  procid.h  setjmp.h  sgstat.h  signal.h
  stdio.h  strings.h  systype.d  time.h

 LIB/……ライブラリ定義ファイル・ディレクトリ
  cio.l  clib.l  clibn.l  cstart.r  math.l
  sys.l  usr.l

 SYS/……システム・ファイル・ディレクトリ
  Errmsg  Motd  Scred.Help  password  termset
```

で開発し，そのつど kernel などのシステム・モジュール群とマージして ROM 化を行い，その動作をテストしながら，各段階を経て最終的な OS9Boot ファイルを完成させます．**図 5** に示した各ディレクトリ，ファイル群と完成したブート・ファイルをともにシステム・ディスク上に書き込み，X68000 IPL ROM の形式にのっとった IPL をトラック 0 セクタ 1 に書き込みます．

▶メモリ配置

最終的なターゲット・システムのメモリ配置を**図 6** に示します．また，起動直後のモジュール配置は**図 4** のとおりです．

このメモリ配置は，ブートストラップ・プログラムに一任されており，ターゲット・システムの個性を活かせるようになっています．

vect 領域は 0 番地からロング・ワード×$100 個ぶんです．ジャンプ・テーブルはその後に 10 バイト×254 個ぶんです．

その後の領域は未使用の RAM 領域としてカーネルに報告しています．意味はまったくない無駄な領域です．$2000 からの $400 バイトは，ディスクの IPL プログラムが読み込まれる領域です．

SSP の領域は $3000 からで，$3000 から $1000 バイトはシステム・グローバル領域です．

$4000 からの $10000 バイトは ROM として定義され，この領域には各種のモジュールがマージされた形で保管されています．この領域の情報をブート・プログラムからスタックに積まれて渡されたカーネルは，自分を含めどんなモジュールがあるか調べに行きます．

〔図 6〕メモリ配置

アドレス	内容	サイズ
$00000000	Vector Table	$400
$400	Jump Table	$9EC
$DEC	未使用	
$2000	IPL(boot)	$25E
$225E	Init SSP ↑	
$3000	システム・グローバル	$1000
$4000	ROM (OS9Boot) kernel clock pipeman pipe ⋮	$10000
$14000	free RAM	$EC000
$FFFFF		

このときそれらのモジュールは連続している必要はありません．つまり，このターゲット・システムでは移植段階でさまざまなモジュールを組み合わせるため余裕をもって 64 K バイト用意しています．そこに新しいモジュールを書き込んでブートに制御を移すだけでテストを行うことができます．特別な追加作業は必要ありません．$14000 からはすべてフリー RAM 領域です．

▶ブート・プログラム

ブート・プログラムは OS-9 のモジュール形式を取る必要はありません(シャープの OS-9/X68K では "rom" という名前のモジュールとしてブート・ファイルの先頭に位置している)．

このブート・プログラムはカーネルに飛び込む前，つまり OS-9 が起動する前のさまざまなハードの状態をチェックし初期化を行って，OS-9 の起動環境を設定する役目をもっています．

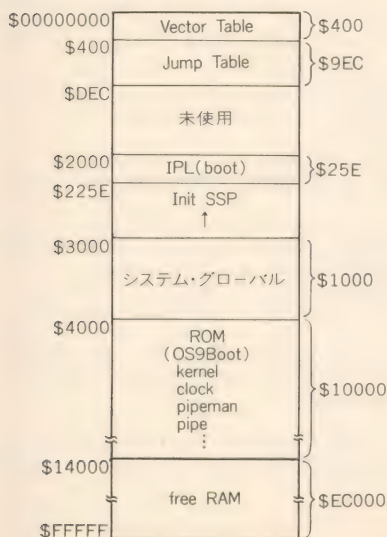
今回の移植では，ブートの役目であるメモリ・チェックなどの機能を省き(IPL ROM で行っているであろうから)，基本的な部分だけで構成しました．最終的にはトラック 0 に書き込まれる IPL プログラムの中に組み込みました．詳しくはシステム・ディスクの解説で行います．

▶仮想 ROM 化

ディスク・ベースで立ち上がるシステムも，あらかじめ ROM に必要なモジュールを書き込んであるシステムも，基本的には同じと考えられます．違う点はブート・プログラムからカーネルに飛び込む前までの状態と，起動後の sysgo の働きのみです．

つまり，ディスク・ベース・システムは，リセット後 IPL ROM によってディスク上の IPL プログラムを呼び出し，実行させ OS 本体(OS9Boot ファイル：必要最低限のモジュール群)をメモリ上に配置します．ここからは ROM 化と同様で，OS9Boot ファイルが格納されている領域を RAM であってもかまわずに ROM として定義(カーネルに通知)します．それ以降は，この領域で書換えは行われません．

移植作業ではこれを利用し，ターゲットの ROM Debugger で RAM 上に必要最大限のモジュール群をマージした OS9Boot ファイルを読み込んで，仮想的に ROM 化 OS-9 を形成してブート・プログラムに制御を移し起動させてテストしました．

このように OS-9 は，ディスク・デバイスを用いるか用いないかの違いだけで，起動時での ROM ベースとディスク・ベース・システムとが区別されているといえます．

また，今回の記事程度のノウハウを知っていればユーザ独自のシステム構成を実現できるでしょう．実際，筆者は0.8Mバイトの ROM(0.2Mバイトの ROM ディスクを含む)と0.2Mバイトの SRAM(バッテリ・バックアップ RAM ディスク)にシステムを構築し，いっさい FDD をアクセスさせずに立ち上がるフル・システム(FD を含む)を利用しています．

▶デバッグ

今回の移植ではシステム・レベル・デバッガや ICE を使用しませんでした．したがって，新たに作成したデバイス・ドライバのデバッグには苦労しました．ドライバの実行はスーパバイザ・モードで行われます．これを利用し，クロック・モジュールなどの割込みをレベル 7 ですべてマスクし，ブート・プログラム中のキャラクタ出力を用いて，ドライバの各所にマクロ・コマンドを埋め込み，メッセージを端末に出力させて，むりやりデバッグしました．

▶ ROM 化第 1 段階(端末での shell)

では，順を追って移植作業を大まかに解説していきます．

始めは OS-9 を走らせることだけに専念します．必要最低限の無修正のモジュールはつぎのとおりです．

- kernel(核)
- cio(CI/O トラップ・ハンドラ)
- scf(SCF マネージャ)
- math(演算トラップ・ハンドラ)
- pipeman(パイプ・ファイル・マネージャ)
- pipe(デバイス・デスクリプタ)
- nil(デバイス・デスクリプタ)
- null(デバイス・ドライバ)

新たに作るモジュールは，

- init(システム・パラメータ)
- sysgo(カーネルを除いた最初のプロセス・プログラム：ProcsID＝2.)
- clock(クロック・ドライバ)
- scc(RS-232-C ドライバ)
- t0(RS-232-C デスクリプタ)

これらに加えてブート・プログラムを作り，起動確認用につぎのモジュールをいっしょにマージして OS9Boot ファイルを作ります．

- shell(コマンド・インタプリタ)
- mdir(メモリ・モジュール表示)
- procs(プロセス表示)
- mfree(メモリ使用状況表示)

新たに作るモジュールの詳細は，後述のソース説明で行います．これで OS-9 を起動すれば最初のプロセスである sysgo が子プロセス(PID＝3)shell を起動し shell のスタンプ・メッセージとともにプロンプトが端末上に現れます．そこで mdir などを起動させ確認します．また，nil にリダイレクトしてマルチタスクを試すのもよいでしょう．

▶ ROM 化第 2 段階(ディスク・ドライバ)

第1段階ではモジュールのテストを行う環境を作りました．これだけでは拡張性がありません．そこでファイル・システムを構築します．ファイル・システムが完成すれば，ターゲット単体(端末を含む)で自身を開発できます．第1段階に加えて作成するモジュールはつぎのとおりです．

- disk(FD デバイス・ドライバ)
- d0(FDD 0 デバイス・デスクリプタ)
- d1(FDD 1 デバイス・デスクリプタ)

これらに加えて，つぎのコマンド・モジュールも必要です．

- iniz(デバイスの追加)
- dir(ディレクトリ表示)
- load(モジュールのロード)

これらをブート・ファイルにマージするか，ROM 領域の空いている場所に読み込んでテストします．テストの始めは iniz と dir で読込みができるかどうかを試します．できるようならリダイレクトや copy などを load して書込みをテストします．最後に format コマンドを実行してみて動作すれば移植作業の山は越えたことになります．

FD デバイス・ドライバが完成したら sysgo や init を書き換えて CMD, DEFS, SYS などのディレクトリを書き込んだ仮システム・ディスクを用意してもよいでしょう．

▶ ROM 化第 3 段階(vterm)

これまでは scc 端末デバイス・ドライバを標準入出力と定義(init 内)してきましたが，ターゲット本体には立派すぎるほどのヒューマン・インターフェースが備わっています．そこでそれらを一部利用したターミナル・デバイス・ドライバを作成します．新しく作るモジュールはつぎのとおりです．

- vterm(ターミナル・デバイス・ドライバ)
- term(ターミナル・デバイス・デスクリプタ)

これらをシステムに加える方法は，disk モジュールと同様です．これをテストする場合には，iniz を用いて vterm を登録し，scc 端末から新たに shell をつぎのように起動します．

```
$ shell </term >>>/term
```

このように標準入力，標準出力，標準エラー出力を

termにリダイレクトし,子プロセスとして起動させます.

これまで利用してきたscc端末デバイス・ドライバはキャラクタ入出力に割込みを用いていません. したがってshellの入出力プロセスにおいてsleepはありえません. いったんsccドライバの入力ルーチンに入れば, 入力されるまでマルチタスクは止まってしまいます. つまり真のマルチタスクは実現できないことになります.

これが完成したら同時にsysgo, initも書き換える必要があります. 標準入出力をtermに設定し直さなければならないためです.

ところで, sccを使ってマルチユーザを実現したい場合には, sccデバイス・ドライバを本当のドライバに作り直す必要があります. つまり割込みサービス・ルーチンを作り, ハードウェアの待ち時間をsleep状態で過ごすようにすることです. これについてはvtermを参考にするか, 参考文献1)のaciaデバイス・ドライバを参考にするとよいでしょう.

▶システム・ディスク完成(IPL)

最後にパワーONで立ち上がるシステム・ディスクを作って移植完了です.

図4に示したこれらのモジュールをすべてマージして最終的なブート・ファイルOS9Bootを作ります. これをフォーマットしたてのルート・ディレクトリに書き込みます. **図4**に示したモジュールの他にも必要なモジュールや使用頻度の高いモジュールをいっしょにマージしてもかまいません. 本来はcioやshellなどのモジュールは将来の変更やOSの使用目的, メモリ効率によってリンク形態が変わるので, 通常実行ディレクトリ内に格納しておくべきものです.

つぎに**図3**に示したとおりのディレクトリやモジュール, ファイルなどをCOPYして, 最後にROM DebuggerでIPLを書き込みます.

ここでX68000のIPLについて触れておきます. リセット後IPL ROMはSRAMの内容からブート情報を読み出し, それが標準であればトラック0の第1セクタから$400(1024)バイトを$2000番地に読み込んできます. そして読み込んだIPLプログラムの第1バイトが$60であるかどうか確認し, その命令にジャンプします. つまりIPLの最初の命令はブランチ命令でなくてはなりません. このブランチは, ブランチでもワード・ブランチでもかまいません. また, X68000のシステム・ディスク(立上げディスク)のトラック0のセクタ長は何でもよいようです.

このプログラムの冒頭ではX68000のIOCSコールを使ってブート・ファイルを読み出しています. この部分はIPLの機能そのもので, その後はブート・プログラムへと続きます. また, これをOS-9のリロケータブル・アセンブラ, リンカで作る場合にはリンク時に生のコードを出力させるため－r(aw：生)オプションを指定します. ブート・プログラムの機能の詳細については, 先月号の「ROM化の技法」についての記事を参照してください.

システム・リセットはハードウェアの初期設定(SCCの初期化), ベクタ・テーブル・セットアップ, ジャンプ・テーブルのセットアップ, システム・グローバル・エリアのクリアを行い, 最後にレジスタにブート情報をセットアップして, RAMリスト, ROMリストを作りカーネルにジャンプします.

キャラクタ入出力は起動時のカーネルとエラー・メッセージ出力などに使われ, ユーザのデバッグにも使われます.

ここまでの構成でOS-9としては十分動作しますが, OS-9の設計思想である「なるべく外部記憶はアクセスしない」ということからも, RAMディスクは必要であり, X68Kには512Kバイトものグラフィック RAMとテキストRAMがあり, これを遊ばせておくのはもったいないことです. そこでこれらを利用してRAMディスクを作りました.

4 ソース・プログラム

移植作業の各段階ではinitモジュールを始めとしてsysgo, IPLなどをそのつど書き換えてきましたが, ここではとくに断わらないかぎり, 最終的な(といっても, じつはまだまだな)ソースについて解説します. 各モジュールの一般的な機能などは他の書籍に譲ります.

● **init**(**リスト1**；p.187)

起動時にカーネルが参照する定義データです.

後方に位置するストリング・テーブルは, 各移植段階で何回か書き直しました. 移植第1段階ではコンソール・ネームを/t0とし, システム・デバイスは用いていませんので, SysDev自体を0とします(ストリング・テーブルはあってもかまわない).

余談ですが, initで定義しているsysgoやclockのモジュール・ネームは変更できます. 組込み制御関係ではsysgoの代わりに開発したモジュールを設定することもあるでしょう. また, 各テーブル・サイズやスライス・タイムをいろいろ変えてみるのもおもしろいかもしれません.

● **clock**(**リスト2**；p.187)

マルチタスクや日時計測のためのタイマ・ドライバです. initルーチンではTimerDをセットアップして割込みを登録しスタートさせます. その後, TimerDか

ら割込みがかかるたびに，サービス・ルーチンでつぎの割込みまでの時間をセットして帰ります．タイマを再設定しないと割込みは発生しつづけますが，256までカウントするのでチック間隔が延びてしまいます．

前述したとおり，このタイマにはMFPを利用しますが，その割込みはCPUがMFPにベクタを要求した時点で解除される自動割込み終了モードを用いています．

● **sysgo(リスト3**；p.188)

起動後最初のプロセスとして起動されるユーザ・プログラムです．このプログラムは標準的な作業を行っています．ワーキング実行ディレクトリをCMDSに設定し，子プロセスとしてshellを起動しています．shellが終了すると再びshellを起動するループになっています．このプログラムをユーザなりに書き換えることによってさまざまなシステムを構築できます．シャープのOS-9/X68Kのsysgoがだいぶ大きいのは，X68000のハードを活かしたさまざまな機能を実現するためでしょう．これを参考にするのも勉強の一つです．

● **disk(リスト4**；pp.188-194)

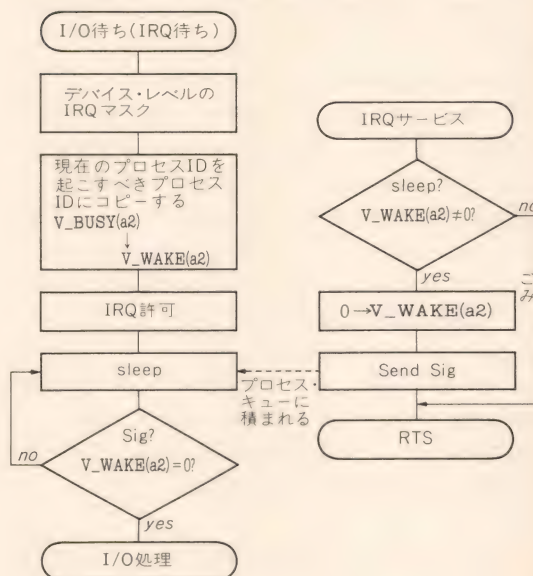
ディスク・ドライバの役目は，RBFから渡されたパス・デスクリプタとLSNから，物理的なセクタをアクセスすることです．また，ハード的な待ち時間を無駄にしないように，そのプロセスをsleep状態にし，割込みを利用して再び実行状態に戻す必要があります．

ここで重要なポイントは，このsleepにあります．どんなデバイスのドライバでも重要なことは，そのデバイスの制御はもちろんですが，そのデバイスの特性から，どこでsleepをかけ，またどのようにsleepすれば良いか，sleepする時間がどの程度なのかという問題です．sleepの仕組みを**図7**で簡単に示します．今回示したソース・リストを参考にするとよいでしょう．なお，このdiskデバイス・ドライバとvtermデバイス・ドライバは，マイクロウェアのポート・パックに含まれるrb765.a，sc6850.aを参考に作りました．したがって，手を加えた部分，新たに挿入した部分が入り乱れていますので，例として最善であるとはいえませんが，sleepの方法やデバイスの制御などのポイントは参考になると思われます．

プログラムの詳細は，ソース・リストの各ラインに挿入したコメントで大体わかることと思います．また，ドライバの各ルーチンの処理内容や呼出しにともなうレジスタの意味については，本誌別稿あるいは他の文献を参考にしてください．

このプログラムで特筆すべき点は以下のとおりです．

(1) IDセクタのキャッシング

LSN 0(IDセクタ)にかぎりライト・スルーの疑似キャッシュを設けました．これは一つの試みであり，さらにアロケーション・ビットマップ，ルート・ディレクトリなどへの拡張を想定しています．イジェクト・マスクによりライト・バックも可能でしょう．

〔図7〕sleepのしくみ(デバイス・ドライバ内)

(2) TimerAによるモータ制御

モータのシャットOFFにTimerAを用いてディスク・アクセス時の回転安定待ち時間をなくしました．X68000ではモータ制御をソフトで行います．また，アクセス・ドライブは一時に一つとかぎられています．したがって，このFDCの特徴である複数台ドライブ・シークは行うことができません．

(3) DMAの他チャネルの禁止

DMACのチャネル0を使用していますが，他のチャネルは使用禁止としています．今回のシステムでは，とくに他のドライバでDMA転送を必要としていません．しかし，他の理由でDMAを用いたシステムを構成する場合には，このドライバを書き換える必要があります．その場合には，DMACが長期にわたりバスを占有しないようにする必要があります．

(4) パス・デスクリプタの無視

このプログラムはパス・デスクリプタ(とくにデバイス・デスクリプタの情報)をだいぶ無視した形を取りました．通常は，デバイス・デスクリプタ中のパラメータがパス・デスクリプタなどにコピーされ，ドライバはそれを参照して各種制御を行いますが，今回は，2HDのメディアしかアクセスしないこととしたため，そのようなパラメータをドライバ内にもち，それを参照してアクセスするようにしています．本来は2DD,

異種形式などへの配慮が当然必要でしょう.

以上の点が注意事項です. この他にもシーク関係で問題があるらしくseek, recalibrateパターン($EEEE)を数回繰り返してしまう問題があります.

デバイス・デスクリプタ(**リスト5**;p.195)については問題ありません. ソース・リストにはd0しか示しませんでしたが, d1についてはオプション領域のドライブ番号を変え, リンク時に−n(ame)オプションでモジュール名を変更するだけで済みます. dd(デフォルト・ドライブ)についてはd0のソースをそのまま利用し, リンク時に名前を変更するだけです.

● **ramdsk(リスト6**;pp.195-196)

基本的にはdiskと同じです. 制御するデバイスがないため割込みもDMAも用いていません. かえって, RBFデバイス・ドライバの働きがわかりやすいと思われます. とくに注意すべき点を解説します.

(1) フォーマッティング

リセット前後で内容を変化させないようにするために, RAMディスクをリンクした後にフォーマットする必要があります. メディア的にはハード・ディスクとしています.

(2) RAMディスクの容量

r0はグラフィックRAMの512Kバイト, r1はテキストRAMをvtermと分割するため384Kバイト(T0, T1, T2)の容量をもちます. これらはドライバの中で定義していますが, 本来はデバイス・デスクリプタから得るべき内容です. これを改良してユーザ独自のドライバを作成するのもよいでしょう(ちなみにver 1.2で作ったこのRAMディスクがver 2.0で動作している).

デバイス・デスクリプタ(**リスト7**;p.196)は, フォーマットのためにセクタ数のところとドライブ番号を変えるだけで, 後はd0, d1, ddと同じです.

● **vterm(リスト8**;pp.196-203)

vtermデバイス・ドライバは, キーボードからMFPに送られてきたシリアル・データを割込みサービス・ルーチンでバッファリングし, リード・ルーチンでそのバッファのデータを1文字ずつ取り出します. 割込みルーチンでキーボードから受け取ったキャラクタがキーボード割込みなどのコードだった場合には, sleep状態のプロセスにシグナルを送ります.

ライト・ルーチンではd0レジスタに与えられたキャラクタ・コードのフォントをCG-ROMから読み出し, テキストRAM上に出力しています. とくにsleepが必要なデバイスは用いていません. 注意する点はつぎのとおりです.

(1) キーボードからの入力

キーボードからの入力はMFP内のUSARTを用いています. このシステムで日本語はいっさい扱っていませんので, キーボードの機能はSHIFT, CTRL, CAPSのみです. したがって, そのインジケータであるLEDはCAPSしか対応していません.

(2) テキストRAMとスクロール

テキストRAMはプレーン3のみを用いています. したがって, 1色しか対応できません. また, スクロールにはX68K特有のラスタ・コピーを用いています. このラスタ・コピー・ルーチンでは割込みをすべて禁止しているため, OS-9らしからぬドライバとなってしまいました. このあたりをsleepなりDMAなどで作り直す必要があります. そのためにはラスタ・コピーの性質を深く調べなければならないでしょう.

(3) カーソルのブリンク

カーソル・ブリンクはリード・ルーチンをコールしたプロセスのsleepを無限ではなく0.5秒間隔で設定しON, OFFさせています.

(4) パス・デスクリプタの無視

このプログラムもまたパス・デスクリプタを部分的に無視した形を取りました. したがって, tmodeなどのプログラムで設定される値のいくつか(タブ長など)は無視されます. このあたりは行き当たりばったりの域を出ないところでしょう.

デバイス・デスクリプタterm(**リスト9**;p.203)ではとくに注意する点はありません.

ところで, このvterm中にRTCをアクセスして実時間をシステム・グローバル領域に設定しています. これはいただけない話で, 本来はclockかsysgoに組み込むべきものです.

最後に

今回の移植システムは, 目標を辛うじて達成したものの, そのできばえはまだまだです. しかし, この経験からOS-9の魅力とその奥義をかいまみることができ多くのことを学びとることができました.

移植の機会を与えてくださった吉岡良雄氏, 環境整備に協力してくださった阿部英志氏, ともに移植に携わった山川直巳氏, 激励をくださった星勝徳氏に誌面を借りてお礼を申し上げます.

参考文献

1)「OS-9/68000の研究」,『マイコンピュータ』, No.25
2)『OS-9/68000テクニカルマニュアル』, 秀和システム
3)『X68000テクニカルデータブック』, アスキー出版
4)「FDC, DMAC」,『トランジスタ技術』, 1988年10月号
5)「SCC」,『プロッセッサ』, 1985年12月号

かんの・さとし 岩手大学

〔リスト1〕init モジュール

```
        nam init
        ttl config_module
        use <defsfile>
***************************************************************
*       init module for ROM system

Edition         equ     5
Typ_Lang        set     (Systm<<8)+0
Attr_Rev        set     (ReEnt<<8)+0
                psect   init,Typ_Lang,Attr_Rev,Edition,0,0

**************************************************
* config constants (OS-9/68000 Level One Ver1.2)

CPUTyp  set     68000   cpu type
Level   set     1       OS-9 level One
Vers    set     1       version 1
Revis   set     2       revision 2
Edit    set     0       edition
Site    set     0       installation site code
MDirSz  set     128     module directory size
PollSz  set     32      IRQ polling table size
DevCnt  set     32      device table size
Procs   set     64      process table size
Paths   set     64      path table size
Slice   set     2       tick per time slice
SysPri  set     128     initial system priority
MinPty  set     0       system min executable prior
MaxAge  set     0       system max natural age limit
MaxMem  set     0       Top of RAM
Events  set     0       event table size

**************************************************
*       init module table

        dc.l    MaxMem          unused
        dc.w    PollSz          IRQ polling table size
        dc.w    DevCnt          device table size
        dc.w    Procs           process table size
        dc.w    Paths           path table size
        dc.w    0               IOMan name offset(unused)
        dc.w    SysStart        mod nam offset(ex:sysgo)
        dc.w    SysDev          default dev nam offset(ex:/d0)
        dc.w    ConsolNm        std I/O pth nam ofst(ex:/term)
        dc.w    Extens          custom mod name offset
        dc.w    ClockNm         clock mod nam offset(ex:clock)
        dc.w    Slice           tick per time slice
        dc.w    UsrAct          accounting pack name offset
        dc.l    Site            installation site code
        dc.w    MainFram        installation name offset
        dc.l    CPUTyp          cpu type
        dc.b    Level,Vers,Revis,Edit
        dc.w    OS9Rev          OS-9 revision string offset
        dc.w    SysPri          initial system priority
        dc.w    MinPty          init sys min exec priority
        dc.w    MaxAge          max system natural age limit
        dc.l    MDirSz          module directory size
        dc.w    Events          event table size
        dc.w    0,0,0,0,0,0,0,0 reserved
        dc.w    0,0,0,0,0,0,0   reserved

***************************************************************
*       string table

OS9Rev          dc.b    "OS-9Level One V1.2",0
SysStart        dc.b    "sysgo",0
ConsolNm        dc.b    "/term",0
ClockNm         dc.b    "clock",0
MainFram        dc.b    "OS-9/68K for X68000",0
SysDev          dc.b    "/d0",0
Extens          dc.b    "OS9P2",0
UsrAct          dc.b    "UAcct",0

        ends
```

〔リスト2〕clock ドライバ・モジュール

```
        nam clock
        ttl Driver
***************************************************************
*       Clock Driver Module

Typ_Lang        set     (Systm<<8)+Objct
Attr_Rev        set     (ReEnt<<8)+Rev
Rev             set     1
Edit            set     6
Stk             equ     0
                psect   Prog,Typ_Lang,Attr_Rev,Edit,Stk,Init
                use     <defsfile>

***************************************************************
*       constants table

TicksSec        equ     100     100 ticks per second
ClkVect         equ     $44     int timer D
ClkPrior        equ     1       highest
PortAddr        equ     $E88000 MFP port
IntEnablA       equ     $7      IERA offset
IntEnablB       equ     $9      IERB offset
ISRA            equ     $f      in service register A
ISRB            equ     $11     in service register B
IntMskA         equ     $13     IMRA offset
IntMskB         equ     $15     IMRB offset
VectReg         equ     $17     vector register of MFP
IntPrScr        equ     $1d
TimDCnt         equ     $25

***************************************************************
*       program start

Init
        tst.w   D_TckSec(a6)            already init?
        bne.s   ClockExit               branch if so

* -----------------disable IRQ of timer D----------------------
        move.l  #PortAddr,a3            point to MFP
        andi.b  #$f0,IntPrScr(a3)       count stop
        andi.b  #$ef,IntEnablB(a3)      IRQ disable
        andi.b  #$ef,IntMskB(a3)        IRQ mask
        move.b  #$40,VectReg(a3)        set vctr(auto end IRQ)

* --------------set up IRQ fot timer D------------------------
        move.w  #TicksSec,D_TckSec(a6)  set # of ticks/sec
        move.b  #TicksSec,D_Tick(a6)    set # of ticks
        moveq.l #ClkVect,d0             vector
        moveq.l #ClkPrior,d1            priority
        lea     ClkSrv(pc),a0           IRQ service entry
        os9     F$IRQ                   get on the table
        bcs.s   ClockExit               branch if error

* ---------------enable IRQ of timer D-------------------------
        move.b  #$c8,TimDCnt(a3)        20kHz->1/200->100Hz
        ori.b   #$10,IntEnablB(a3)      enable
        ori.b   #$10,IntMskB(a3)        unmask
        move.b  #$7,IntPrScr(a3)        4MHz->1/200->20kHz
ClockExit
        rts
***************************************************************
*               IRQ service routine

ClkSrv
        move.b  #$c8,TimDCnt(a3)        restore counter
        movea.l D_Clock(a6),a0
        jmp     (a0)

        ends
```

〔リスト 3〕sysgo モジュール

```
        nam sysgo
        ttl startup module
        use <defsfile>
***************************************************************
*       sysgo program for ROM system

Edition         equ     1
Typ_Lang        set     (Prgrm<<8)+Objct
Attr_Rev        set     2
Priority        equ     128
                psect   test,Typ_Lang,Attr_Rev,Edition,0,Entry

***************************************************************
*       stack space

        vsect
buff    ds.b    255     sysgo stack space
        ends

***************************************************************
*       intercept routine

Intercpt
        os9     F$RTE
        bcs     Err

***************************************************************
*       program start

Entry
        lea     Intercpt(pc),a0 point to intercept routine
        os9     F$Icpt
        bcs     Err
        lea     CmdStr(pc),a0   point to default exe dir nam
        moveq   #Exec_,d0       execute mode
        os9     I$ChgDir        chd to ex:"CMDS" from root(/d0)
* ------------execute first procedure file--------------------
        moveq   #0,d0           any type module
        moveq   #0,d1           default mamory size
        moveq   #StartSiz,d2    first procedure file name size
        moveq   #3,d3           copy # of conecting I/O path
        move.w  #Priority,d4    medium priority
        lea     ShellStr(pc),a0 primery mod nam offset(shell)
        lea     StartStr(pc),a1 procedure file name offset
        os9     F$Fork          fork shell with startup
        os9     F$Wait          wait, ignore any error
Loop
        moveq   #0,d0           any type module
        moveq   #0,d1           default memory size
        moveq   #1,d2           # of cr code($0d)
        moveq   #3,d3           copy # of conecting I/O path
        move.w  #Priority,d4    medium priority
        lea     ShellStr(pcr),a0 primery mod nam offset(shell)
        lea     CRChar(pcr),a1  parameter offset($0d)
        os9     F$Fork          fork shell
        bcs.s   Err             branch if error
        os9     F$Wait          wait for sending EOF code
        bcs.s   Err             branch if error
        tst.w   d1              no error code?
        beq.s   Loop            loop if so
Err
        os9     F$PErr          print error message
        bra.s   Loop

***************************************************************
*       string table

ShellStr        dc.b    "shell"
CRChar          dc.b    C$CR,0
StartStr        dc.b    "startup",C$CR,0
StartSiz        equ     *-StartStr
CmdStr          dc.b    "CMDS",C$CR,0

        ends
```

〔リスト 4〕disk デバイス・ドライバ・モジュール ①

```
        nam     Disk Driver NEC uPD72065
        ttl     Driver Module for X68000

Edition  equ    11              current edition number
Typ_Lang set    (Drivr<<8)+Objct  Device Driver In Assembly
Attr_Rev set    (ReEnt<<8)+0
Stack    equ    $100
         psect  Prog,Typ_Lang,Attr_Rev,Edition,Stack,DiskEnt
         use    <defsfile>

* ----------------------Static Storage definitions-----------
                vsect
                ds.b    drvs2.1 (=$de)
V_BUF           ds.l    64      Addr of local buffer
V_LSN           ds.l    1       logical sector #
V_IMask         ds.w    1       Interrupt Mask Value
V_Side          ds.b    1       Side select value
V_Sector        ds.b    1       sector buffer
V_Track         ds.b    1       track buffer
V_CurDrv        ds.b    1       Drive select bit
V_MOTOR         ds.b    1       motor ready bits
V_LSNS          ds.b    1       LSN0 buffer valid bits
                align
V_LSN00         ds.l    64      LSN0 buffer of drive 0
V_LSN01         ds.l    64      LSN0 buffer of drive 1

* -------------------------Nec command buffers----------------
Command1        ds.b    1
Command2        ds.b    1
Command3        ds.b    1
Command4        ds.b    1
Command5        ds.b    1
Command6        ds.b    1
Command7        ds.b    1
Command8        ds.b    1
Command9        ds.b    1

* -------------------------Nec result buffers-----------------
Results         ds.b    9
                ends

* ----------------------X68000 register layouts---------------
MSR             equ     1       NEC765 main status register
DataReg         equ     3       NEC765 Data Register
Option          equ     5       Drive status Register
Media           equ     $80     Media insert
MskEject        equ     $40     mask eject switch
MotorCtl        equ     7       5 1/4 motor ctrl 0= motor off
MTRON           equ     $80     motor on
IntEnabl        equ     $e9c001 IRQ enable/disable 0=disable
IntVecNm        equ     $e9c003 IRQ vector register

* --------------------IntEnable register bits----------------
IM.PRT  equ     %00001110 Printer IRQ mask pattern
IM.FDD  equ     %00001101 Floppy Disk Drv IRQ mask pattern
IM.FDC  equ     %00001011 Floppy Disk Drv IRQ mask pattern
IM.HDD  equ     %00000111 Hard Disk Drv IRQ mask pattern

* -------------------------Nec 72065 Commands-----------------
F.Specfy        equ     $03     specify command
F.Rest          equ     $07     Restore cmd
F.Seek          equ     $0F     Seek cmd
F.ReadSc        equ     $46     Read sector
F.WrtSec        equ     $45     Write sector
F.WrtTrk        equ     $4D     Write track
F.SnsDrv        equ     $04     sense drive status
F.SnsIRQ        equ     $08     sense interrupt status
N               equ     $01     256 bytes/sector
HLT             equ     $10     Head load time
Filler          equ     $E5     sector fill byte
GPL8DD          equ     $36     gap length for format
GPL             equ     $E      gap length for 5&8" drives
DTL             equ     $FF     data length n/a for 256byte sec
```

```
EOTDD8          equ     $1A     last sect on track 8" dd
DelayTim        equ     30      time to delay between commands

* -----------------------Main Status Register bits------------
DOB      equ     $1      drive zero in seek mode
D1B      equ     $2      drive one in seek mode
D2B      equ     $4      drive two in seek mode
D3B      equ     $8      drive three in seek mode
CB       equ     $10     read or write in progress
NDM      equ     $20     fdc in non dma mode
DIO      equ     $40     0 = processor > fdc
RQM      equ     $80     data register ready
CB_Bit   equ     4       FDC busy
DIO_Bit  equ     6       data input/output
RQM_Bit  equ     7       request for master
Invalid  equ     $80     invalid command code

* ----------------Nec Error register bits(result status)------

* --------------------------------ST0------------------------
IC              equ     $C0     command completion status
Seek_Bit        equ     5       seek end flag bit
EC              equ     $10     Fault or Bad Restore
NR              equ     $08     device not ready

* --------------------------------ST1------------------------
EN       equ     $80     end of cylinder
DE       equ     $20     CRC error
ND       equ     $04     seek err or
NW       equ     $02     write protect
MA       equ     $01     missing address (seek error)

* --------------------------------ST3------------------------
RY_Bit   equ     5       ready signal from FDD

* --------------------------Error code bits------------------
MA_Bit   equ     0       missing address mark
NW_Bit   equ     1       disk write protected
ND_Bit   equ     2       no data
NR_Bit   equ     3       not ready
EC_Bit   equ     4       equipmentceck
DE_Bit   equ     5       data error (crc error)

* -----------------------Bit numbers for DD_FMT--------------
Side_Bit        equ     0       0=single 1=double
Dens_Bit        equ     1       0=single 1=double
Trks_Bit        equ     2       0=48 tpi 1=96 tpi

* ------------------68901 MFP registor layouts---------------
MFP             equ     $e88000 MFP port address in X68000
M.IERA          equ     $07     int enable register A
TIMA_Bit        equ     5       Timer A
M.VECT          equ     $17     vector register
M.TACR          equ     $19     Timer A control register
TAECM           equ     $08     event count mode
M.TADR          equ     $1f     Timer A data register
TDMOFF          equ     $80     delay for shut off motor(2.3s)

* ---HD63450 DMAC registor layouts par channel without D.GCR---
DMAC     equ     $e84000 DMAC port address in X68000
D.CSR    equ     $00     channel status register
COC_Bit  equ     7       channel operation complete
D.CER    equ     $01     no use
D.DCR    equ     $04     device control register
D.OCR    equ     $05     operation control register
D.SCR    equ     $06     sequence control register
D.CCR    equ     $07     channel 0 control register
D.CCR1   equ     $47     channel 1 control register
D.CCR2   equ     $87     channel 2 control register
D.CCR3   equ     $c7     channel 3 control register
SAB      equ     $10     software abort
DMAStrt  equ     $80     start operation
D.MTC    equ     $0a     memory transfer counter
D.MAR    equ     $0c     memory address register
D.DAR    equ     $14     device address register
```

```
D.BTC   equ     $1a     no use
D.BAR   equ     $1c     no use
D.NIV   equ     $25     no use
D.EIV   equ     $27     no use
D.MFC   equ     $29     memory fuction code register
D.CPR   equ     $2d     channel priority register
D.DFC   equ     $31     device function code register
D.BFC   equ     $39     no use
D.GCR   equ     $ff     don't care !

* -----------------Branch Table--------------------
DiskEnt
        dc.w    InitDisk    Initialize i/o
        dc.w    ReadDisk    Read sector
        dc.w    WritDisk    Write sector
        dc.w    GetStat     Get status
        dc.w    PutStat     Put status
        dc.w    Term        Terminate Device

*************************************************************
*               Initialize
*************************************************************
InitDisk
        movea.l  V_PORT(a2),a3       point to FDC ports
        andi.b   #IM.FDC,IntEnabl    FDC IRQ disable
        andi.b   #IM.FDD,IntEnabl    FDD IRQ disable
* ----------------------initialize drive tables-------------
        moveq    #2,d0               set loop count
        move.b   d0,V_NDRV(a2)       Number of Drives
        move.b   #3,V_CurDrv(a2)     Init high drive #
        lea      DRVBEG(a2),a0       Point At First Table
Init10
        move.l   #$000f701a,(a0)     Set Up Size
        move.w   #$ffff,V_TRAK-DRVBEG(a0) set high track #
        lea      DRVMEM(a0),a0       Move To Next Table
        subq.b   #1,d0               last drive?
        bne.s    Init10              branch if not
        clr.b    V_MOTOR(a2)         init motor rdy bits
        clr.b    V_LSNS(a2)          init LSNO buf valid bit
* --------------------------initialize DMAC-----------------
        bsr      DMAini              DMA device initialize
* --------------------------initialize FDC------------------
        move.b   #F.Specfy,Command1(a2) put last command in buf
        move.b   #$d0,Command2(a2)   set SRT(3ms) & HUT(0ms)
        move.b   #$10,Command3(a2)   HLT(16ms) ND(DMA mode)
        moveq    #3,d4               load command count
        moveq    #0,d7               clear transfer mode
        bsr.     DoComand            process the command
        bcs.s    BadUnit             exit if error
* ------------------------Set up for IRQ's------------------
        moveq   #0,d0
        move.b  d0,d2
* ------------------------Set up for IRQ of FDC-------------
        move.b  M$Vector(a1),d0 IRQ vector # from descriptor
        move.b  d0,IntVecNm     IRQ controler PRN,FDD,FDC,HDC
        move.b  M$IRQLvl(a1),d2 get hardware IRQ level
        lsl.w   #8,d2           shift to IRQ mask
        bset    #SupvrBit+8,d2  set system state bit
        move.w  d2,V_IMask(a2)  save for future use.
        move.b  M$Prior(a1),d1  get polling priority
        lea     IRQSrvc(pcr),a0 Point To IRQ Routine
        OS9     F$IRQ           Get On The Table
* ------------------------Set up for IRQ of FDD Eject-------
        move.b  M$Vector(a1),d0 IRQ vector # from descriptor
        addi.b  #1,d0           FDD (FDC>FDD>HDD>PRT)
        move.b  #1,d1           set priority
        lea     IRQDRV(pc),a0   point to IRQ routine
        os9     F$IRQ
* ------------------------Set up for IRQ of Timer A---------
        movea.l #MFP,a3                 point to MFP port
        bclr.b  #TIMA_Bit,M.IERA(a3)    timer A IRQ disable
        bset.b  #TIMA_Bit,M.IMRA(a3)    timer A IRQ mask
        move.b  #$40,M.VECT(a3)         set MFP vector
        move.b  #$4d,d0                 set timer A vector
        move.b  #1,d1                   set priority
```

〔リスト 4〕disk デバイス・ドライバ・モジュール ②

```
        lea     IRQTimeA(pc),a0         point to IRQ routine
        os9     F$IRQ
Return
        rts                             exit
BadUnit
        move.w  #E$Unit,d1              set UNIT error code
        ori     #Carry,ccr              set carry
        rts                             exit with error

*****************************************************************
*               Write Sector
*****************************************************************
WritDisk
        move.l  d2,-(a7)        save logical sector #(number)
        bne.s   Write10         branch if not writing sect 0
        btst    #FmtDis_B,PD_Cntl+1(a1) ok to write sect 0
        bne.s   Write99         no goto error rpt routine
Write10
        moveq   #F.WrtSec,d3    write a sector cmd
        moveq   #2,d7           flag disk write
        movea.l PD_BUF(a1),a5   point to buffer
        bsr     XfrSec          transfer sector
        movem.l (a7)+,d2        restore sector #
        bcs.s   WritErr         Leave If Error
        tst.b   PD_VFY(a1)      Verify ?
        bne.s   WritExit        No, Leave
        move.l  d2,-(a7)        save sector #
        lea     V_BUF(a2),a5    point to verify buffer
        bsr     ReadDs10        Re-Read The Written Block
        movem.l (a7)+,d2        restore sector #
        bcs.s   VerifyEr        exit with error
        lea     V_BUF(a2),a5    point to verify buffer
        movea.l PD_BUF(a1),a0   point to original buffer
        move.w  #256/4,d0       get # of loop count
        bra.s   Verify10
VerifyLp
        subq.w  #1,d0           is loop count 0?
        beq.s   WritExit        branch if so
Verify10
        cmpm.l  (a0)+,(a5)+     is data the same?
        beq.s   VerifyLp        branch if so
VerifyEr
        bra.s   WritDisk        rewrite to success
WritExit
        tst.l   d2              LSN0?
        bne.s   WritExit9       branch if not
        move.b  PD_DRV(a1),d1   get # of drive is 1?
        bne.s   WritExit1       branch if so
        lea     V_LSN00(a2),a0  point to LSN0 of drive 0 buffer
        bra.s   WritExit2
WritExit1
        lea     V_LSN01(a2),a0  point to LSN0 of drive 1 buffer
WritExit2
        movea.l PD_BUF(a1),a5   point to original buffer
        move.w  #256/4-1,d0     get # of loop count
WritExitl
        move.l  (a5)+,(a0)+     data copy
        dbra    d0,WritExitl
        bset.b  d1,V_LSNS(a2)   set valid flag of LSN0 buffer
WritExit9
        moveq   #0,d1           No Errors
WritErr
        rts                     exit
Write99 lea     4(a7),a7        restore stack ptr
        move.w  #E$Format,d1    set error code
        ori     #Carry,ccr      flagwrite error
        rts                     exit

*****************************************************************
*               Read Sector
*****************************************************************
ReadDisk
        movea.l PD_BUF(a1),a5   point to store buffer
        tst.l   d2              reading LSN0?
        bne.s   ReadDs10        branch if not

        move.b  PD_DRV(a1),d0   get # of drive
        btst.b  d0,V_LSNS(a2)   LSN0# buffer valid?
        beq.s   ReadDs10        branch disk read routin if not

        lea     V_LSN00(a2),a0  point to LSN00
        move.w  #256/4-1,d1     get # of loop count
        tst.b   d0              access drive 0?
        beq.s   ReadDs1         branch if so
        lea     $100(a0),a0     point to LSN01
ReadDs1
        move.l  (a0)+,(a5)+     data copy
        dbra    d1,ReadDs1
        bra.s   Read25
ReadDs10
        moveq   #F.ReadSc,d3    get NEC read command
        moveq   #1,d7           flag disk read
        tst.l   d2              reading sector 0?
        bne.s   XfrSec          branch if not then exit
        bsr.s   XfrSec          read sector 0 will return
        bcs.s   ReadDs99        exit if error
Read20
        movea.l PD_BUF(a1),a3   point to store buffer
        lea     V_LSN00(a2),a0  point to LSN00 buffer
        move.w  #256/4-1,d1     get # of loop count
        move.b  PD_DRV(a1),d0   get # of drive is 0?
        beq.s   Read1           branch if so
        lea     $100(a0),a0     point to LSN01 buffer
Read1
        move.l  (a3)+,(a0)+     data copy
        dbra    d1,Read1
        bset.b  d0,V_LSNS(a2)   set flag of LSN0#
Read25
        movea.l PD_DTB(a1),a0   point to store buffer
        movea.l PD_BUF(a1),a3   point to read buffer
        move.w  #DD_SIZ-1,d1    Copy This Many+1
Read30
        move.b  (a3,d1.w),(a0,d1.w) copy data
        dbra    d1,Read30       branch if not
        moveq   #0,d1           clear carry
ReadDs99
        rts                     exit

*****************************************************************
*               Transfer Sector
*****************************************************************
XfrSec
        move.l  d2,V_LSN(a2)    buffer LSN
        movea.l V_PORT(a2),a3   get address of FDC port
        move.w  #E$NotRdy,d1    set error code
        bsr     MotorON         stabilize motor
        bcs     SectEr10        branch if error
        bsr     Wait            wait for request by FDC
        move.w  #%1110111011101110,d6 recal,retry pattern
        bra.s   XfrSec20
XfrSec10
        bsr     Restore         recalibrate to track zero
        bcs     SectEr10        branch if error
XfrSec15
        move.l  V_LSN(a2),d2    restore LSN
XfrSec20
        bsr     Select          get drive table pointer
        bcs.w   SectEr10        branch if error
        move.l  DD_TOT(a0),d0   get total # of sectors
        lsr.l   #8,d0           adjust for 3 byte value
        cmp.l   d2,d0           sector out of range?
        bls     SectErr         branch if so
        moveq   #0,d0
        move.b  d0,V_Track(a2)  clear track number
        moveq   #0,d5           clear all of d5
        tst.l   d2              LSN0?
        beq.s   XfrSec40        if branch so
```

```
        move.b  DD_TKS(a0),d5   get # of sectors per trak
        beq     BadUnit         exit with error
        divu    d5,d2           find track #
        lsr.w   #1,d2           adjust track number
        bcc.s   XfrSec40        branch if side 0
        move.b  #4,V_Side(a2)   set side flag
XfrSec40
        addq.w  #1,d2           adjust track # for system track
        move.b  d2,V_Track(a2)  set track #
        swap    d2              get sector # in lower word
        addq.w  #1,d2           adjust sector # for starting 1
        move.b  d2,V_Sector(a2) set sector #
        bsr     SetTrk          move head to new track
        bcs.s   XfrSec60        branch if error
        move.w  #$100,d5        set trancefer count
        bsr     SetUp           set up command buffer
        bcs.s   SectEr10        exit with error
        bsr     DoComand        do transfer
        bcc.s   XfrSec70        branch if no error
XfrSec60
        cmpi.b  #1,PD_Trys(a1)  retry flag?
        beq.s   XfrSecEr10      branch if no retry
        lsr.w   #1,d6           shift recal or retry bit
        bcc.s   XfrSec10        branch if recalibrate
        bne.s   XfrSec15        branch if retry
        bra.s   SectEr10
XfrSec70
        movem.l a3,-(a7)        save FDC port address
        lea     DMAC,a3         get DMAC port address
        btst.b  #COC_Bit,D.CSR(a3) DMA complete?
        beq.s   XfrSecErrr      branch if not
        move.b  #$ff,D.CSR(a3)  reset all bits of CSR
        movem.l (a7)+,a3        restore FDC port add
        bsr     MotorOFF        set up motor off timer
        moveq   #0,d1           clear carry
        rts                     exit or return without error
SectErr
        move.w  #E$Sect,d1      flag sector out of range
SectEr10
        movem.l a3,-(a7)        save FDC port address
        lea     DMAC,a3         point to DMAC port
XfrSecErrr
        move.b  #SAB,D.CCR(a3)  stop DMA channel 0
        move.b  #$ff,D.CSR(a3)  reset all bits of CSR
        movem.l (a7)+,a3        restore FDC port add
        bsr     MotorOFF        set up motor off timer
        ori     #Carry,ccr
        rts                     exit or return with error

*************************************************************
*               Motor ON
*************************************************************
MotorON
        movem.l a3,-(a7)        save FDC port address
        movea.l #MFP,a3         point to MFP address
        bclr.b  #TIMA_Bit,M.IERA(a3) disable IRQ of Timer A
        movem.l (a7)+,a3        restore FDC port address
        move.b  PD_DRV(a1),d1   get # of drive
        btst.b  d1,V_MOTOR(a2)  motor on already?
        bne.s   Motor30         branch if so
        bset    d1,Option(a3)   active option sig
        cmpi.b  #Media,Option(a3) Media insert?
        bne.s   Motor25         branch if not
        ori.b   #MTRON,d1       set motor on bit
        move.b  d1,MotorCtl(a3) motor on

        move.w  #8,d1           set retry # of motor stabilize
Motor10
        btst.b  #DIO_Bit,MSR(a3) dir is CPU -> FDC ?
        beq.s   Motor20         branch if so
        btst.b  #RQM_Bit,MSR(a3) data request by FDC?
        beq.s   Motor10         branch if not request
        tst.b   DataReg(a3)     dummy read
        bra.s   Motor10         test again
Motor20
        bsr     Wait            wait for request by FDC
        move.b  #F.SnsDrv,DataReg(a3) send first cmd to FDC
        bsr     Wait            wait for request by FDC
        move.b  PD_DRV(a1),DataReg(a3) send second cmd to FDC
        bsr     Wait            wait for request by FDC
        move.b  DataReg(a3),d0  get result ST3
        btst.b  #RY_Bit,d0      motor ready for access?
        bne.s   Motor30         branch if so
        subi.b  #1,d1           count stabilize check
        bne.s   Motor10         branch if retry
Motor25
        move.w  #E$NotRdy,d1    error code set
        ori     #Carry,ccr      set error flag
        rts                     return whith error
Motor30
        move.b  PD_DRV(a1),d0   get # of drive
        bset.b  d0,Option(a3)   active option sig
        move.b  #MskEject,Option(a3) mask eject switch
        clr.b   V_MOTOR(a2)     clear flag of each drive
        bset.b  d0,V_MOTOR(a2)  set flag of motor stabilize
        rts                     return with no error

*************************************************************
*               MotorOFF
*************************************************************
MotorOFF
        move.b  PD_DRV(a1),d0   get # of drive
        bset.b  d0,Option(a3)   active option sig
        move.b  #$00,Option(a3) cancel mask of eject
        movem.l a3,-(a7)        save FDC port
        movea.l #MFP,a3         point to MFP
        move.b  #TDMOFF,M.TADR(a3) delay time to shut off motor
        move.b  #TAECM,M.TACR(a3) set TACR to event count mode
        bset.b  #TIMA_Bit,M.IERA(a3) enable IRQ of Timer A
        movem.l (a7)+,a3        resoter FDC port
        rts                     return

*************************************************************
*               DMAini
*************************************************************
DMAini
        movem.l a3,-(a7)        save FDC port
        lea     DMAC,a3         point to DMAC
        move.b  #SAB,D.CCR3(a3) stop DMA channel 3
        move.b  #SAB,D.CCR2(a3) stop DMA channel 2
        move.b  #SAB,D.CCR1(a3) stop DMA channel 1
        move.b  #SAB,D.CCR(a3)  stop DMA channel 0
        move.b  #$ff,D.CSR(a3)  reset all bits of CSR
        move.b  #$80,D.DCR(a3)  set DCR to following...
*non-hold cycle steal,68k peliphelal,8bits device,non-IRQ inPCL
        move.b  #$04,D.SCR(a3)  set SCR to following...
*MAR is increment mode,DAR is non-count mode-------------------
        move.b  #$05,D.MFC(a3)  set MFC to following...
*superviser mode for data memory access------------------------
        move.b  #$00,D.CPR(a3)  set CPR to following...
*channel priority is hieghest----------------------------------
        move.b  #$05,D.DFC(a3)  set DFC to following...
*superviser mode for device access-----------------------------
        movea.l (a7),a4         get FDC port address
        addq.l  #DataReg,a4     adjust device address
        move.l  a4,D.DAR(a3)    set device address to DAR
        movem.l (a7)+,a3        restore FDC port
        rts


*************************************************************
*               Command buffer Setup
*************************************************************
SetUp
        movem.l d0,-(a7)        save
        move.b  d3,Command1(a2) move command
        move.b  V_Side(a2),d0   get # of side (%00000H00)
        or.b    d0,Command2(a2) merge with drive #(by Select)
        move.b  V_Track(a2),Command3(a2) set up track #
```

〔リスト 4〕disk デバイス・ドライバ・モジュール ③

```
        lsr.b   #2,d0           move again for side register
        move.b  d0,Command4(a2) set head number
        move.b  V_Sector(a2),Command5(a2) set up sector #
        move.b  #N,Command6(a2) set up bytes per sector
        move.b  V_Sector(a2),Command7(a2) say last sect/track
        move.b  #GPL,Command8(a2) set up gap length
        move.b  #DTL,Command9(a2) set up sectot length
        moveq   #9,d4           set command size
        movem.l (a7)+,d0        restore

* --------------------fall through to transfer data------------

****************************************************************
*               DMA set up
****************************************************************
DMASet
        bsr     DMAini          initialize FDC
        movem.l a3,-(a7)        save FDC port
        lea     DMAC,a3         point to DMAC
        move.b  #$ff,D.CSR(a3)  reset all bits of CSR
        btst    #0,d7           transfer direction
        beq.s   DMASw           branch if write
        move.b  #$b2,D.OCR(a3)  set OCR to following...
*FDC->Mem,n-BTD,n-pack,8b-port,byte trans,n-chain,REQG=REQline
        bra.s   DMASs
DMASw
        move.b  #$32,D.OCR(a3)  set OCR to following...
*Mem->FDC,n-BTD,n-pack,8b-port,byte trans,n-chain,REQG=REQline
DMASs
        move.l  a5,D.MAR(a3)    set MAR point to buffer
        move.w  d5,D.MTC(a3)    set MTC
        move.b  #DMAStrt,D.CCR(a3) DMAC START
        movem.l (a7)+,a3        restore FDC point
        rts

****************************************************************
*               Restore Drive to Track Zero
****************************************************************
Restore
        bsr.s   Select          SELECT DRIVE
        bcs.s   Restor20        branch if error
        move.b  #10,V_Track(a2) seek out ten tracks
Restor10
        bsr.s   SetTrk          execute seek
        bcs.s   Restor10        branch if error
        move.b  #F.Rest,Command1(a2) buffer command
        clr.b   Command3(a2)    looking for track 0
        moveq   #2,d4           set # of command bytes
        movem.w d7,-(a7)        save transfer mode
        moveq   #0,d7           no transfer data
        bsr     DoComand        issue seek command
        movem.w (a7)+,d7        restore trasfer mode
        bcs.s   Restor10        branch if error
        move.w  #0,V_TRAK(a0)   clear buffer on success
Restor20
        rts                     with no error or select error
ErrNtRdy
        move.w  #E$NotRdy,d1    set error code
        ori     #Carry,ccr      flag error
        rts                     return with error

****************************************************************
*               Select Drive
****************************************************************
Select
        move.b  #0,V_Side(a2)   set side zero
        movea.l PD_DTB(a1),a0   point to drive table
        move.b  PD_DRV(a1),d0   Get Logical Unit Number
        cmp.b   V_CurDrv(a2),d0 same drive as before?
        beq.s   Select30        branch if so
        cmp.b   V_NDRV(a2),d0   drive in range?
        bhs.s   BadDrive        branch if so
        move.b  d0,V_CurDrv(a2) update drive #
Select30
        move.b  d0,Command2(a2) save drive #
        rts
BadDrive
        move.w  #E$Unit,d1      flag bad unit
        ori     #Carry,ccr
        rts

****************************************************************
*               Step Head to New Track
****************************************************************
SetTrk
        move.b  V_Track(a2),d0  get # of track
        cmp.b   V_TRAK(a0),d0   same track?
        beq.s   SetTrk20        branch if so
SetTrk10
        move.b  #F.Seek,Command1(a2) set command buffer
        move.b  d0,Command3(a2) buffer track #
        moveq   #3,d4           set command count
        movem.w d7,-(a7)        save transfer mode
        moveq   #0,d7           no transfer data
        bsr.s   DoComand        issue seek command
        movem.w (a7)+,d7        restore transfer data
        bcs.s   SetTrk20        branch if error
        move.b  V_Track(a2),V_TRAK(a0) set # of new track
SetTrk20
        rts

****************************************************************
*               Wait for controller ready
****************************************************************
Wait
        bsr.s   Delay   wait for valid status
Wait20
        tst.b   MSR(a3) ready for command?
        bpl.s   Wait20  branch if not
        rts


****************************************************************
*Delay 12 Micro Seconds for controller to give valid status
****************************************************************
Delay
        movem.l d0,-(a7)        save
        moveq   #DelayTim,d0    set loop count
Delay10
        subq.b  #1,d0           is count 0?
        bpl.s   Delay10         branch if not
        movem.l (a7)+,d0        restore
        rts

****************************************************************
*               Issue Transfer Commands
****************************************************************
DoComand
        movem.l a0-a6/d7,-(a7)  savem all
        moveq   #50,d0          try fifty times
DoCmnd10
        btst    #DIO_Bit,MSR(a3) device ready for commands?
        beq.s   DoCmnd40        branch if so
        tst.b   DataReg(a3)     ready byte of data
        bsr.s   Delay           wait before testing again
        dbra    d0,DoCmnd10     try again
DoCmnd20
        move.w  #E$NotRdy,d1    exit with error
        movem.l (a7)+,a0-a6/d7  restore
        ori.w   #Carry,sr       flag error
        rts     exit            with error
DoCmnd40
        btst    #CB_Bit,MSR(a3) still execution mode?
        bne.s   DoCmnd20        branch if so
        move.b  MSR(a3),d0      get controllor status
        andi.b  #(D0B!D1B!D2B!D3B),d0 any devices in seek mode
        beq.s   DoCmnd50        branch if not
```

```
        move.b  #F.SnsIRQ,DataReg(a3) sense IRQ status
        bsr.s   Wait            wait for request
        tst.b   DataReg(a3)     get first sense byte
        bsr.s   Wait            wait fot request
        tst.b   DataReg(a3)     get last sense byte
        bra.s   DoCmnd40
DoCmnd50
        lea     Command1(a2),a4 point to command buffer
        subq.b  #2,d4           adjust loop count
DoCmnd80
        bsr.s   Wait            wait for valid status
        move.b  (a4)+,DataReg(a3) send next command
        dbra    d4,DoCmnd80     branch untill one byte left
        bsr.s   Wait            wait for valid data
        tst.b   d7              transfer data?
        beq     IRQCmnd         branch if not
        move.b  (a4)+,d1        get last byte
        move    sr,-(a7)        save IRQ status
        move    V_IMask(a2),sr  mask IRQs
        move.w  V_BUSY(a2),V_WAKE(a2) set up for interrupt
        move.b  d1,DataReg(a3)  move last command
        ori.b   #$04,IntEnabl   enable FDC IRQs
        move    (a7)+,sr        enable IRQs
DoCmnd90
        moveq   #0,d0   sleep forever until send sig of wake
        os9     F$Sleep
        tst.w   V_WAKE(a2)      valid wakeup?
        bne.s   DoCmnd90        branch if not
TfrDone
        movem.l (a7)+,a0-a6/d7  restore

* -------------------fall through to read results------------

***************************************************************
*               Read Results
***************************************************************
ReadRslt
        movem.l d0/a0,-(a7)     save
        lea     Results(a2),a0  point to result buffer
        bra.s   ReadRs40
ReadRs10
        move.b  DataReg(a3),(a0)+ move data to buffer
ReadRs40
        bsr     Wait            wait for controller ready
        btst    #DIO_Bit,MSR(a3) still reading data(resultST#)?
        bne.s   ReadRs10        branch if so

* -------------------------test for errors------------------
ErrorTst
        move.b  Results(a2),d0  get ST0
        andi.b  #(IC!EC!NR),d0  strip all but error bits
        beq.s   No_Error        exit with no errors
        move.w  #E$NotRdy,d1    flag not ready error
        btst    #(NR_Bit),d0    device not ready?
        bne.s   Error_Ex        branch if so
        btst    #(EC_Bit),d0    bad equipment?
        bne.s   Error_Ex        branch if so
        move.b  Results+1(a2),d0 get next result byte
        move.w  #E$Seek,d1      flag seek error
        lsr.b   #1,d0           seek error?
        bcs.s   Err_Ex          branch if so
        btst    #(ND_Bit-1),d0  seek error?
        bne.s   Error_Ex        branch if so
        move.w  #E$WP,d1        flag write protect error
        lsr.b   #1,d0           write protect?
        bcs.s   Err_Ex          branch if so
        move.w  #E$DevBsy,d1    flag device busy
        lsr.b   #3,d0
        move.w  #E$CRC,d1       flag crc error
        lsr.b   #1,d0           crc error?
        bcs.s   Err_Ex          branch if so
        move.w  #E$Unit,d1      catch all error
Error_Ex
        ori     #Carry,ccr      set carry
        movem.l (a7)+,a0/d0     restore
```

```
        rts
No_Error
        moveq   #0,d1           clear carry
        movem.l (a7)+,a0/d0
        rts                     exit with no error

***************************************************************
*       Issue Last Command from command buffer using interrupts
***************************************************************
IRQCmnd
        move.b  (a4)+,d1        get last command
        movem.l (a7)+,a0-a6/d7  restore(from DoComnd)
        cmpi.b  #F.Specfy,Command1(a2) Specify command?
        bne.s   IRQCm10         branch if not
        move.b  d1,DataReg(a3)  move last command
        moveq   #0,d1           no errors
        rts                     exit
IRQCm10
        move    sr,-(a7)        save IRQ status
        move    V_IMask(a2),sr  mask IRQs
        move.w  V_BUSY(a2),V_WAKE(a2) set up for interrupt
        move.b  d1,DataReg(a3)  move last command
        ori.b   #$04,IntEnabl   enable FDC IRQs
        move    (a7)+,sr        enable IRQs
IRQCm20
        moveq   #0,d0           sleep forever
        os9     F$Sleep
        tst.w   V_WAKE(a2)      valid wakeup?
        bne.s   IRQCm20         branch if not

* ----------------fall through to sense what caused the IRQ---

***************************************************************
*               Sense Irq
***************************************************************
SenseIRQ
        andi.b  #IM.FDC,IntEnabl FDC IRQ disable
        move.b  #F.SnsIRQ,DataReg(a3) give controller command
        bsr     Wait            wait for result STatus 0
        move.b  DataReg(a3),d4  read ST0
        cmpi.b  #Invalid,d4     is this command valid?
        beq.s   SenseIRQ        branch if not
        bsr     Wait            wait for Present Cylinder #
        move.b  DataReg(a3),d2  read PCN
        btst    #Seek_Bit,d4    was seek complete
        beq.s   Sens_Err        branch if not
        cmp.b   Command3(a2),d2 seek to right track?
        bne.s   Sens_Err        branch if not
        moveq   #0,d1           no error
        rts
Sens_Err
        move.w  #E$NotRdy,d1
        ori     #Carry,ccr
        rts

***************************************************************
*               GetStat/PutStat
***************************************************************
PutStat
        movea.l V_PORT(a2),a3   point to FDC port
        cmpi.w  #SS_WTrk,d0     is it a Write Track call?
        beq.s   WriteTrk        branch if so
        cmpi.w  #SS_Reset,d0    is it a restore call?
        bne.s   GetStat         branch if so
        move.w  #E$NotRdy,d1    flag not ready
        bsr     MotorON         stabilize motor
        bcs.s   PutStatEr       branch if error
        bsr     Restore         recalibrate head
        bcs.s   PutStatEr       branch if error
        bra     MotorOFF        shut off motor then exit
GetStat
        move.w  #E$UnkSvc,d1    flag unknown service code
PutStatEr
        ori     #Carry,ccr      flag error
        rts
```

〔リスト 4〕disk デバイス・ドライバ・モジュール ④

```
***************************************************************
*               WriteTrk
***************************************************************
WriteTrk
        btst    #FmtDis_B,PD_Cntl+1(a1) enable for formatting
        beq.s   WrtTrk10        branch if so
        move.w  #E$Format,d1    flag bad mode
        ori     #Carry,ccr      flag error
        rts                     exit with error
WrtTrk10
        movea.l V_PORT(a2),a3   point to FDC
        bsr     MotorON         stabilize motor
        bcs     WTrkEr10        branch if error
        bsr     Select          select proper drive
        bcs     WTrkEr10        exit with error
        movea.l PD_RGS(a1),a4   get register pointer
        move.b  R$d2+3(a4),V_Track(a2) save track # for seek
        bsr     SetTrk          seek to track
        bcs     WTrkEr10        branch if error
        move.b  #F.WrtTrk,Command1(a2) write ID
        move.w  R$d3+2(a4),d3   get format byte
        move.b  d3,DD_FMT(a0)   move format byte
        btst    #Side_Bit,d3    is it side 0?
        beq.s   WrtTrk30        branch if so
        move.b  #1,V_Side(a2)   set to side 1
        bset    #2,Command2(a2) set to head 1
WrtTrk30
        move.b  #N,Command3(a2) set up # bytes per sector
        move.b  #EOT8DD,Command4(a2) get sectors/track
        move.b  #GPL8DD,Command5(a2) set up gap length
        move.b  #Filler,Command6(a2) set filler byte
        move.b  R$d2+3(a4),d0   get track #
        move.b  V_Side(a2),d1   get side #
        moveq   #N,d3           get # bytes/sector
        move.w  #EOT8DD,d4      get # of sectors/track
        subq.w  #1,d4           adjust for loop count
        lea     V_BUF(a2),a5    build track buffer
        movea.l R$a1(a4),a6     get interleave table pointer
WrtTrk40
        move.b  d0,(a5)+        set cylinder #
        move.b  d1,(a5)+        set head #
        move.b  (a6)+,(a5)      get record #
        addi.b  #1,(a5)+        adjust recod #
        move.b  d3,(a5)+        set # of bytes/sector
        dbra    d4,WrtTrk40
        moveq   #6,d4           get # of command bytes
        moveq   #2,d7           set transfer mode to write
        lea     V_BUF(a2),a5    point to track buffer
        move.w  #26*4,d5        trancefer count set
        bsr     DMASet          DMA Set up
        bsr     DoComand        execute the command
        bcs.s   WTrkEr10        branch if error
        bsr     MotorOFF
        moveq   #0,d1           no error
        rts
WTrkEr10
        bsr     MotorOFF
        ori     #Carry,ccr
        rts

***************************************************************
*               Terminate use of device
***************************************************************
Term
        movea.l V_PORT(a2),a3   get port address
        andi.b  #IM.FDC,IntEnabl FDC IRQ disable
        andi.b  #IM.FDD,IntEnabl FDD IRQ disable
        bclr.b  #TIMA_Bit,M.IERA(a3) timer A IRQ disable
        bset.b  #TIMA_Bit,M.IMRA(a3) timer A IRQ mask
        bsr.w   IRQTimeA        shut off motor
        bset.b  #0,Option(a3)   active option sig
        move.b  #$00,Option(a3) cancel mask of eject
        bset.b  #1,Option(a3)   active option sig
        move.b  #$00,Option(a3) cancel mask of eject
* -------------------------------take FDC---------------------
        movea.l V_PORT(a2),a3   get port address
        move.b  M$Vector(a1),d0 get vector #
        suba.l  a0,a0           take device off table
        OS9     F$IRQ
* -------------------------------take FDD---------------------
        movea.l V_PORT(a2),a3   get port address
        move.b  M$Vector(a1),d0 get vector #
        addq.b  #1,d0           adjust for FDD
        suba.l  a0,a0           take device off table
        OS9     F$IRQ
* -------------------------------take Timer A-----------------
        movea.l V_PORT(a2),a3   get port address
        move.b  #$4d,d0         get vector # of timer A
        suba.l  a0,a0           take device off table
        OS9     F$IRQ
        rts

***************************************************************
*               Interrupt Service routine
***************************************************************
* -----------------------interrupt from FDC------------------
IRQSrvc
        andi.b  #IM.FDC,IntEnabl FDC IRQ disable
        move.w  V_WAKE(a2),d0   was driver waiting?
        bne.s   IRQSr20         branch if so
        btst    #DIO_Bit,MSR(a3) ready for command
        bne.s   IRQExit         branch if not
        move.b  #F.SnsIRQ,DataReg(a3) issue Sense Int St CMD
        bsr     Delay           wait 12 us
        tst.b   DataReg(a3)     read first byte
        bsr     Delay           wait 12 more us
        tst.b   DataReg(a3)     read second byte
        bra.s   IRQExit
IRQSr20
        clr.w   V_WAKE(a2)      flag IRQ occured
        moveq   #S$Wake,d1      get wake up signal
        os9     F$Send          send driver signal
IRQExit
        moveq   #0,d1
        rts
* -----------------------interrupt from MFP------------------
IRQTimeA
        movea.l #MFP,a3         point to MFP port
        bclr.b  #TIMA_Bit,M.IERA(a3) timer A IRQ disable
        move.b  #$00,MotorCtl(a3) shut off motor 0
        move.b  #$01,MotorCtl(a3) shut off motor 1
        clr.b   V_MOTOR(a2)     clear each dfv flag
        moveq   #0,d1
        rts
* -----------------------interrupt from FDD------------------
IRQDRV
        andi.b  #IM.FDD,IntEnabl FDD IRQ disable
        moveq   #0,d0           init # of drive
IRQDRV0
        bset.b  d0,Option(a3)   active option sig
        cmpi.b  #Media,Option(a3) Media insert?
        beq.s   IRQDRV1         branch if so
        bclr.b  d0,V_LSNS(a2)   clear flag of LSN0 buff
        bclr.b  d0,V_MOTOR(a2)  clear flag of motor stabilize
IRQDRV1
        addi.b  #1,d0           next drive
        cmpi.b  #2,d0           end of test each drive?
        bne.s   IRQDRV0         branch if not
        ori.b   #$02,IntEnabl   enable FDD IRQ
        moveq   #0,d1
        rts

        ends
```

〔リスト5〕d0 デバイス・デスクリプタ・モジュール

```
 nam d0
 ttl Device Descriptor for Floppy disk controller
 use <defsfile>

Edition equ 5 current edition number

Single   equ 0
Double   equ 1
Five     equ 0
Eight    equ 1
Hard     equ $80
ON       equ 1
OFF      equ 0

****************
* Descriptor Defaults
Mode     set Dir_+ISize_+Exec_+Updat_
BitDns   set Single
Heads    set 2
StepRate set $0c
Intrleav set 3
NoVerify set OFF
DnsTrk0  set Double
DMAMode  set 0 non dma device
SegAlloc set 8 minimum segment allocation size
TrkOffs  set 1
SectOffs set 1

DiskKind set Eight
Cylnders set 77
BitDns   set Double
TrkDns   set Single
SectTrk  set $1a
SectTrk0 set $1a
DevCon   set 0

Density  set BitDns+(TrkDns<<1)
DiskType set DiskKind+(DnsTrk0<<5)

SectSize set 256 default sector size  256 bytes.
FmtEnabl set 0 enable formatting
FmtDsabl set 1 disable formatting
Control  set FmtEnabl enable formatting

Trys     set 7 number of Trys

TypeLang set (Devic<<8)+0
Attr_Rev set (ReEnt<<8)+0
 psect d0,TypeLang,Attr_Rev,Edition,0,0
 dc.l $e94000 port address
 dc.b $80      auto-vector trap assignment
 dc.b 1        IRQ hardware interrupt level
 dc.b 1        irq polling priority

 dc.b Mode    device mode capabilities
 dc.w FileMgr file manager name offset
 dc.w DevDrv  device driver name offset
 dc.w DevCon  (reserved)
 dc.w 0,0,0,0 reserved
 dc.w OptLen

* Default Parameters
OptTbl
 dc.b DT_RBF  device type
 dc.b 0 drive number
 dc.b StepRate step rate
 dc.b DiskType type of disk 8"/5"/Hard
 dc.b Density  Bit Density and track density
 dc.w Cylnders-TrkOffs number of cylinders
 dc.b Heads Number of Sides (Floppy) Heads(Hard Disk)
 dc.b NoVerify OFF = disk verify ON = no verify
 dc.w SectTrk default sectors/track
 dc.w SectTrk0 default sectors/track track 0
 dc.w SegAlloc segment allocation size
 dc.b Intrleav sector interleave factor
 dc.b DMAMode DMA mode (none)
 dc.b TrkOffs track base offset
 dc.b SectOffs sector base offset
 dc.w SectSize # of bytes/sector
 dc.w Control format control byte
 dc.b Trys number of retrys 0 = no retrys/error correction
OptLen equ *-OptTbl

FileMgr dc.b "RBF",0 Random block file manager
DevDrv dc.b "disk",0
 ends
```

〔リスト6〕ramdsk デバイス・ドライバ・モジュール ①

```
        nam ramdsk
        ttl Ram disk driver
        use     <defsfile>
***************************************************************
*       device driver module for ramdisk

Edition         equ     1
Typ_Lang        set     (Drivr<<8)+Objct
Attr_Rev        set     (ReEnt<<8)+1
Stk             equ     $100
        psect   Prog,Typ_Lang,Attr_Rev,Edition,Stk,Entry

***************************************************************
*       static storage requirements

V_RBF           equ     $de

                vsect
                ds.b    V_RBF
Sect_Add        ds.l    1       sector address pointer buffer
                ends

***************************************************************
*       branch tables
Entry
        dc.w    Init
        dc.w    Read
        dc.w    Write
        dc.w    GetStat
        dc.w    PutStat
        dc.w    Term

***************************************************************
*       initialize

Init
        lea     DRVBEG+DD_TOT(a2),a0  get drive table pointer
        move.l  #$80001,(a0)          # of sec & sec/track(G-RAM)
        lea     DRVMEM(a0),a0         get next drv tbl ptr
        move.l  #$60001,(a0)          # of sec & sec/track(T-RAM)
        move.b  #2,V_NDRV(a2)         set # of drive
        move.w  #$0316,$e80028        set memory & disp mode
        moveq   #0,d1                 set no error code
        rts                           exit

***************************************************************
*       terminate
Term
        moveq   #0,d1   no error
        rts

***************************************************************
*       GetPut

PutStat
        cmpi.w  #SS_WTrk,d0     format code?
        beq.s   PutStt90        exit with no error if so
        cmpi.w  #SS_Reset,d0    recalibrate code?
        beq.s   PutStt90        exit with no error if so
GetStat
        move.w  #E$UnkSvc,d1    ignore other code
```

〔リスト 6〕ramdsk デバイス・ドライバ・モジュール ②

```
        ori     #Carry,ccr
        rts
PutStt90
        moveq   #0,d1
        rts

*****************************************************************
*       read

Read
        bsr.s   XfrSec          calculate sector address
        bcs.s   Read90          leave if error
        movea.l PD_BUF(a1),a0   point to buffer
        movea.l Sect_Add(a2),a3 point to sector
        move.w  #256/4-1,d0     loop count set
Read10
        move.l  (a3)+,(a0)+     read
        dbra    d0,Read10
        tst.l   d2              read LSN0?
        bne.s   Read30          exit if not
        movea.l PD_DTB(a1),a0   point to device table
        movea.l Sect_Add(a2),a3 point to sector
        move.w  #DD_SIZ-1,d1    get copy byts
Read20
        move.b  (a3,d1.w),(a0,d1.w) copy
        dbra    d1,Read20
Read30
        moveq   #0,d1
Read90
        rts

*****************************************************************
*       write

Write
        bsr.s   XfrSec          calculate sector address
        bcs.s   Write90         leave if error
        movea.l PD_BUF(a1),a0   point to buffer
        movea.l Sect_Add(a2),a3 point to sector
        move.w  #256/4-1,d0     set loop count
Write10
        move.l  (a0)+,(a3)+     write
        dbra    d0,Write10
        moveq   #0,d1
Write90
        rts

*****************************************************************
*       transfer from lsn # to Ram address

XfrSec
        tst.b   PD_DRV(a1)          drive 0(G-RAM)?
        bne.s   XfrSec1             branch if drive 1(T-RAM)
        cmpi.l  #$00000800,d2       LSN < $800?
        bhs.s   SecErr              branch if so
        move.l  d2,d0
        lsl.l   #8,d0               get sector pointer(*$100)
        addi.l  #$00c00000,d0       add G-RAM base address
        move.l  d0,Sect_Add(a2)     buffer
        moveq   #0,d1
        rts
XfrSec1
        cmpi.l  #$00000600,d2       LSN < $600?
        bhs.s   SecErr              branch if so
        move.l  d2,d0
        lsl.l   #8,d0               get sector pointer(*256)
        addi.l  #$00e00000,d0       add T-RAM base address
        move.l  d0,Sect_Add(a2)     buffer
        moveq   #0,d1
        rts
SecErr
        move.w  #E$Sect,d1
        ori     #Carry,ccr
        rts
        ends
```

〔リスト 7〕r0 デバイス・デスクリプタ・モジュール

```
 nam r0
 ttl device descriptor for Ramdisk controller
 use <defsfile>

Edition set 1

TypeLang set (Devic<<8)+0
Attr_Rev set (ReEnt<<8)+0
 psect r0,TypeLang,Attr_Rev,Edition,0,0

 dc.l $c00000 port address(G-RAM)
 dc.b 0 unused vector
 dc.b 0 unused hard IRQ #
 dc.b 0 unused priority
 dc.b Dir_+ISize_+Exec_+Updat_ Mode
 dc.w descmgr file manager name offset
 dc.w descdrv device driver name offset
 dc.w 0 reserved
 dc.w 0,0,0,0 reserved
 dc.w OptLen
OptTbl
 dc.b DT_RBF dvice type
 dc.b 0 drive number
 dc.b 0 step rate(unused)
 dc.b $80 hard disk type
 dc.b 0 density(unused)
 dc.w 1 # of cylinder
 dc.b 1 # of head
 dc.b 0 verify on
 dc.w 2048 # of sector per track
 dc.w 2048 # of sector per track0
 dc.w 8 segment allocation size
 dc.b 0 interleav(unused)
 dc.b 0 DMA(unused)
 dc.b 0 track offset(unused)
 dc.b 0 sector offset(unused)
OptLen equ *-OptTbl

descmgr dc.b "rbf",0
descdrv dc.b "ramdsk",0

 ends
```

〔リスト 8〕vterm デバイス・ドライバ・モジュール ①

```
        nam vterm
        ttl Device driver for X68000
*****************************************************************
*Note:
* font = 16(X)*8(Y)
* screen = 32(chr)*96(chr)
* use area = T3 plane($e60000-$e6ffff) and ($e70000-$e707ff)
*****************************************************************
                use     <defsfile>
Edition         equ     11
Type_Lang set   (Drivr<<8)+Objct
Attr_Rev        set     (ReEnt<<8)+0
        psect   vterm,Type_Lang,Attr_Rev,Edition,512,VtrmEnt

* ---------------------Static storage requirements------------
V_SCF           equ     sctstat.l       (=$54)
BufSiz          equ     $100
                vsect
                ds.b    V_SCF
IRQMask         ds.w    1       IRQ mask value
Next            ds.b    1       pointer of buffer
Count           ds.b    1       counter of rest byte
Brk             ds.b    1       brink sw(bit7) & msk(bit0)flags
RevSw           ds.b    1       charactor reverse switch flag
CurX            ds.b    1       cursor position X
CurY            ds.b    1       cursor position Y
```

```
SESC            ds.b    1       special escape sequence switch
ESCpara         ds.b    1       escape parameter buffer
Cndition        ds.b    1       0,0,0,0,ctrl,shift,capslk,caps
LEDBit          ds.b    1       keyboard LED buffer
SigPrc          ds.l    1       signal buffer
Buff            ds.b    BufSiz  loop buffer pointed by Next(a2)
                ends
* -------------------text area map definitions---------------
T3U     equ     $e60000 T3 plane base address
T3L     equ     $e70000 T3 plane of half base line
* ---------------------CRTC register layouts-----------------
CRTC    equ     $e80000 CRTC base address
C.TSel  equ     $2a     text access & high speed clear plane
C.Ras   equ     $2c     raster of source & destination register
C.Mode  equ     $480    mode register
* -------------------68901 MFP registor layouts--------------
MFP     equ     $e88000 MFP port address in X68000
M.IERA  equ     $07     int enable register A
M.IMRA  equ     $13     IRQ mask register A
M.VECT  equ     $17     vector register
TAECM   equ     $08     event count mode
M.TBCR  equ     $1b     timer B control register
M.TBDR  equ     $21     timer B data register
M.UCR   equ     $29     USART control register
M.RSR   equ     $2b     receive status register
M.TSR   equ     $2d     transmit status register
M.UDR   equ     $2f     USART data register
* -----------RP5C15 Real Time Clock device register layouts----
RTC     equ     $e8a000 RTC port address
R.MODE  equ     $1b     mode register

**************************************************************
*               Execute entry table
**************************************************************
VtrmEnt
        dc.w    Init    initialize I/O
        dc.w    Read    read a charactor
        dc.w    Write   write a charactor
        dc.w    GetStat get status
        dc.w    PutStat put status
        dc.w    TrmNat  terminate device

**************************************************************
*               Init
* Initialize (Terminal) Acia
* Passed: (a1)=device descriptor address
*               (a2)=static storage address
* Returns: cc=carry set if device can't be initialized
* Destroys: (may destroy d0-d7, a0-a5)
**************************************************************
Init
        move    sr,-(a7)        save
        ori.w   #$0700,sr       mask all IRQ

* --------------------initialize static strage---------------
        clr.l   IRQMask(a2)             init static strage area
        move.l  #$01000000,Brk(a2)      enable brink
        move.l  #$000000ff,SESC(a2)     !?
        clr.b   V_ERR(a2)               clear error registor

* --------------line clear for raster copy with nul column----
        move.w  #128*16/4-1,d1  set loop count(font16*8 * 128)
        movea.l #T3L,a0         set T3 half plane base add
Init1
        clr.l   (a0)+           clear
        dbra    d1,Init1        count
* -------------------------clear half T3 plane---------------
        bsr     ClrTxt
* ------------------------initialize txt palettte------------
        move.w  #$7,d1          set loop count
        moveq.l #0,d0           clr palette(%0,T2,T1,T0) data
        lea     $e82200,a0      point to txt palette register
Init2
        move.w  d0,(a0)+        clear
        dbra    d1,Init2        count
        move.w  #$7,d1          set loop count
        moveq.l #$ffff,d0       set palette(%T3,?,?,?) data
Init3
        move.w  #$ffff,(a0)+    set
        dbra    d1,Init3        count

* -----------------MFP reset for keyboard control------------
        movea.l #MFP,a3
        andi.b  #$e0,M.IERA(a3) USART IRQ disable
        move.b  #$40,M.VECT(a3) IRQ vector set
        move.b  #$01,M.TBCR(a3) timer B priscarer set
        move.b  #$0d,M.TBDR(a3) baud rate set
        move.b  #$88,M.UCR(a3)  USART setup
        move.b  #$00,M.RSR(a3)  recieve status register set
        move.b  #$04,M.TSR(a3)  X'fr status register set
        andi.b  #$f7,$e8e007    TV control disable
* ---initialize date and time register of SYSTEM GLOBAL AREA---
        movea.l #RTC,a0         point to RTC
        bclr.b  #0,R.MODE(a0)   select bank 0
        lea     $0c(a0),a1      point to week(time) register
        bsr.s   ReadRTC         get time to d1
        move.l  d1,d0           move time value to d0!
        lea     $1a(a0),a1      point to mode(date) register
        bsr.s   ReadRTC         get date to d1!
        swap    d1
        addi.w  #1980,d1        adjust date(add offset)
        swap    d1
        os9     F$STime
* ----------------Set up for IRQ of Receiver(keyboad)---------
        move.b  #$6,d2          set hardware IRQ level of MFP
        asl.w   #8,d2           shift into priority
        bset    #SupvrBit+8,d2  set system state bit
        move.w  d2,IRQMask(a2)  save for future use
        move.b  #$4c,d0         set vector #
        move.b  #2,d1           set priority
        movea.l #MFP,a3         point to MFP's port address
        lea     IRQSvc(pc),a0   address of IRQ service routine
        OS9     F$IRQ           Add to IRQ polling table
        bcs.s   InitErr         branch if error
*************************************
*disable IRQ
*bit3=receive error
*bit2=transmit buffer empty
*bit1=transmit error
        andi.b  #%11110001,M.IERA(a3)
*************************************
*enable IRQ(bit4=receive buffer full)
        ori.b   #%00010000,M.IERA(a3)
*************************************
*mask IRQ
        andi.b  #%11110001,M.IMRA(a3)
        ori.b   #%00010000,M.IMRA(a3)
        move.b  #$01,M.RSR(a3)  enable Receiver
        move    (a7)+,sr
        moveq   #0,d1           clear error code & flag
        rts

InitErr
        move    (a7)+,sr
        move.w  #E$Unit,d1
        ori     #Carry,ccr
        rts
* -get and add 4 bits BCD code of each number of digits on RTC-
ReadRTC
        moveq   #0,d1   clear for return value
        moveq   #0,d4   clear for calcurate
        moveq   #2,d2   set loop count
Loop
        move.w  -(a1),d3 BCD code(numbers of ten digits=*10)
        andi.w  #$f,d3  strip invalid bits
        move.b  d3,d4   copy for mul by 8 & mul by 2
        lsl.b   #3,d3   multiply got BCD by 8
        lsl.b   #1,d4   multiply got BCD by 2
        add.b   d3,d4   add *2 with *8 is same *10
        move.w  -(a1),d3 BCD code(number of one digit=*1)
```

〔リスト8〕vterm デバイス・ドライバ・モジュール ②

```
        andi.w  #$f,d3  strip invalid bits
        add.b   d4,d3   add *1 & *10 to d3
        asl.l   #8,d1   move for next loop
        or.b    d3,d1   merge two byts for a word
        dbra    d2,Loop next loop?
        rts             return if end
******************************************************************
*               Read
* Return one byte of input from the Acia
* Passed: (a1)=Path Descriptor
*               (a2)=Static Storage address
*               (a4)=current process descriptor
*               (a6)=system global ptr
* Returns: (d0.b)=input char
* cc=carry set, (d1.w)=error code if error
* Destroys: a0
******************************************************************
Read00
        move.w  V_BUSY(a2),V_WAKE(a2) arrange wake up signal
        move    (a7)+,sr        restore IRQs
Read01
        bsr     Brink           brink cursor
        bsr.s   Slep    SLEEP while 0.5 sec or until sendsig
        tst.w   V_WAKE(a2)      valid wakeup?(ret from ACSLER)
        bne.s   Read01          branch if not
Read
        tst.w   SigPrc(a2)      a process waiting for device?
        bne.s   ErrNtRdy        ret dormant terminal error
        move    sr,-(a7)        save current IRQ status
        move    IRQMask(a2),sr  mask irqs
        tst.b   Count(a2)       any data?
        beq.s   Read00          branch if not
        moveq   #0,d0
        move.l  d0,d1
        move.b  Next(a2),d1     get pointer of loop buffer
        lea     Buff(a2),a0     point to loop buffer
        move.b  0(a0,d1),d0     get receive chr
        subi.b  #1,Next(a2)     post decrement
        subi.b  #1,Count(a2)    decrement counter
        move    (a7)+,sr        unmask IRQs
        move.b  V_ERR(a2),PD_ERR(a1) copy I/O status to PD
        beq.s   Read90          return if no error
        clr.b   V_ERR(a2)       for next read
        move.w  #E$Read,d1      signal read error
        ori     #Carry,ccr      return Carry set
Read90
        rts
ErrNtRdy
        move.w  #E$NotRdy,d1
        ori     #Carry,ccr
        rts
******************************************************************
Slep
        movem.l d0/a0,-(a7)
        move.l  #$80000080,d0   sleep for 0.5 sec
        OS9     F$Sleep         wait for input Data
        move.w  P$Signal(a4),d1 signal present?
        beq.s   ACSL90          ..no; return
        cmpi.w  #S$Intrpt,d1    Deadly signal?
        bls.s   ACSLER          ..yes; return error
ACSL90
        btst    #Condemn,P$State(a4) has process died?
        bne.s   ACSLER          ..Yes; return error
        movem.l (a7)+,d0/a0
        rts
ACSLER
        lea     12(a7),a7       Exit to caller's caller
        ori     #Carry,ccr      return Carry set
        rts

******************************************************************
*                                       Write
* Output one character to Acia
* Passed: (d0.b)=char to write
*               (a1)=Path Descriptor
*               (a2)=Static Storage address
*               (a4)=current process descriptor ptr
*               (a6)=system global data ptr
* Returns: none
******************************************************************
Write
        bclr.b  #0,Brk(a2)      mask brink
        bsr     Brink           stop reverse
        andi.l  #$ff,d0         valid data
        cmpi.b  #$20,d0         is it control chr?
        bcs.s   WrtCtrl         branch if so
        tst.b   SESC(a2)        special Escape mode?
        bne     EscExc          branch if so
        bsr     ChrAdd          get charactor address
        bsr     PutChr          display
        tst.b   RevSw(a2)       reverse mode?(by SESC)
        beq.s   Writ20          branch if not
        bsr     ChrAdd          get charactor address
        bsr     ChrRev          reverse charactor
Writ20
        addi.b  #1,CurX(a2)     next X
Writ25
        cmpi.b  #$60,CurX(a2)   over range?
        bcs.s   WritE           branch if not
        clr.b   CurX(a2)        initialize
Writ30
        addi.b  #1,CurY(a2)     next Y
        cmpi.b  #$20,CurY(a2)   over range?
        bcs.s   WritE           branch if not
        move.b  #$1f,CurY(a2)   restore
        bsr     Schroll         schroll up
WritE
        move.b  #$01,Brk(a2)    enable brink
        moveq   #0,d0
        rts                             exit with no error

* ---------------------ASCII control code check---------------
WrtCtrl
        cmpi.b  #$1e,d0         home cursor?
        beq.s   Ctrl1e          branch if so
        cmpi.b  #$1b,d0         ESC code?
        beq.s   Ctrl1b
        cmpi.b  #$1a,d0         clear scleen?
        beq.s   Ctrl1a
        cmpi.b  #$0d,d0         CR?
        beq.s   Ctrl0d
        cmpi.b  #$0c,d0         forward cursor?
        beq.s   Writ20
        cmpi.b  #$0b,d0         up cursor?
        beq.s   Ctrl0b
        cmpi.b  #$0a,d0         down cursor?
        beq.s   Writ30
        cmpi.b  #$09,d0         skip to next tab stop?
        beq     Ctrl09
        cmpi.b  #$08,d0         backward cursor?
        beq     Ctrl08
        bra.s   WritE           other code are not support
Ctrl1e
        clr.b   CurX(a2)        init X
        clr.b   CurY(a2)        init Y
        bra.s   WritE
Ctrl1b
        move.b  #1,SESC(a2)     flag special ESC mode
        clr.b   ESCpara(a2)     init paramtr buf for continue
        bra.s   WritE
Ctrl1a
        bsr     ClrTxt          clear txt
        bra.s   WritE
Ctrl0d
        clr.b   CurX(a2)        init X
        bra.s   WritE
Ctrl0b
        tst.b   CurY(a2)        is it 0?
        beq     WritE           branch if so
```

```
        subi.b  #1,CurY(a2)     cursor up
        bra     WritE
Ctrl09
        andi.b  #$f8,CurX(a2)   back near tab position
        addi.b  #$08,CurX(a2)   add 8
        bra     Writ25          exit for position check
Ctrl08
        subi.b  #1,CurX(a2)     back X
        bpl     WritE           branch if 0
        move.b  #$5f,CurX(a2)   set X for end of line
        subi.b  #1,CurY(a2)     up Y
        bpl     WritE           branch if 0
        clr.b   CurX(a2)        home
        clr.b   CurY(a2)        home
        bra     WritE

* ------------------------ESC parameter check-----------------
EscExc
        tst.b   ESCpara(a2)     continue "ESC=lc"?
        beq.s   Esclc           branch if not
        subi.b  #$20,d0         strip to valid data
        cmpi.b  #1,ESCpara(a2)  second parameter?
        bhi.s   EscX            branch if so
        move.b  d0,CurY(a2)     set Y(first parameter)
        addi.b  #1,ESCpara(a2)  flag for next X
        bra     WritE
Esclc
        cmpi.b  #$3d,d0         direct cursor addressing?
        bne.s   EscM            branch if not
        move.b  #1,ESCpara(a2)  flag parameter continue(X)
        bra     WritE
EscM
        cmpi.b  #$4d,d0         delete line?
        bne.s   EscP            branch if not
        bsr     LinDel          delete line
        bra.s   EscEnd
EscP
        cmpi.b  #$50,d0         delete chr?
        bne.s   EscJ            branch if not
        bsr.s   ChrDel          delete chdr
        bra.s   EscEnd
EscJ
        cmpi.b  #$4a,d0         clear screen?
        bne.s   EscK            branch if not
        bsr     ClrTxt          clear screen
        bra.s   EscEnd
EscK
        cmpi.b  #$4b,d0         clear to end of screen?
        bne.s   EscL            branch if not
        bsr     ClrEoL          clear to end of screen
        bra.s   EscEnd
EscL
        cmpi.b  #$4c,d0         insert line?
        bne.s   Escm            branch if not
        bsr     LinIn           insert line
        bra.s   EscEnd
Escm
        cmpi.b  #$6d,d0         alternate video?
        bne.s   Escn            branch if not
        move.b  #1,RevSw(a2)    alternate video
        bra.s   EscEnd
Escn
        cmpi.b  #$6e,d0         restore normal video?
        bne.s   EscEnd          branch if not
        clr.b   RevSw(a2)       restore normal video
        bra.s   EscEnd
EscX
        move.b  d0,CurX(a2)     set X(last parameter)
EscEnd
        clr.b   ESCpara(a2)     reset flag
        clr.b   SESC(a2)        reset flag
        bra     WritE           end ESC sequense
```

```
****************************************************************
*           put charactor
*       input: d0=chr$,a0=point writed chr
****************************************************************
PutChr
        lsl.l   #4,d0           get charactor offset into ROM
        movea.l d0,a1           move
        adda.l  #$f3a800,a1     point to the charactor
        move.b  (a1)+,(a0)      rom -> txtram
        move.b  (a1)+,$80(a0)
        move.b  (a1)+,$100(a0)
        move.b  (a1)+,$180(a0)
        move.b  (a1)+,$200(a0)
        move.b  (a1)+,$280(a0)
        move.b  (a1)+,$300(a0)
        move.b  (a1)+,$380(a0)
        move.b  (a1)+,$400(a0)
        move.b  (a1)+,$480(a0)
        move.b  (a1)+,$500(a0)
        move.b  (a1)+,$580(a0)
        move.b  (a1)+,$600(a0)
        move.b  (a1)+,$680(a0)
        move.b  (a1)+,$700(a0)
        move.b  (a1),$780(a0)
        rts
****************************************************************
*           charactor delete
****************************************************************
ChrDel
        bsr     ChrAdd          get chr add
ChrDel1
        move.b  1(a0),(a0)      copy lower chr to upper chr
        move.b  $81(a0),$80(a0)
        move.b  $101(a0),$100(a0)
        move.b  $181(a0),$180(a0)
        move.b  $201(a0),$200(a0)
        move.b  $281(a0),$280(a0)
        move.b  $301(a0),$300(a0)
        move.b  $381(a0),$380(a0)
        move.b  $401(a0),$400(a0)
        move.b  $481(a0),$480(a0)
        move.b  $501(a0),$500(a0)
        move.b  $581(a0),$580(a0)
        move.b  $601(a0),$600(a0)
        move.b  $681(a0),$680(a0)
        move.b  $701(a0),$700(a0)
        move.b  $781(a0),$780(a0)
        addi.l  #1,a0           point to next corumn chr
        move    a0,d1           move
        cmpi.b  #$60,d1         last corumn?
        bcs.s   ChrDel1         branch if not
        subi.l  #1,a0           point to last corumn chr
        bsr.s   ClrChr          space
        rts
****************************************************************
*           clear character
*       input: nothing
*       output: nothing
* destroy: d0.l
*       calls: PutChr
****************************************************************
ClrChr
        move.l  #$20,d0 set space code
        bsr     PutChr  display space
        rts
****************************************************************
*           clear to end of line from current cursor X
*       input: notihng
*       output: nothing
* destroy: a0.l,a1.l,d1.l,a5.l
*       calls: ChrAdd,ClrChr
****************************************************************
ClrEoL
        bsr.s   ChrAdd          get chr add
```

続 OS-9/68K徹底解剖(上級編)

〔リスト 8〕vterm デバイス・ドライバ・モジュール ③

```
ClrEoL1
        bsr.s   ClrChr          print space
        lea     1(a0),a0        next corumn
        move.l  a0,d2           move
        cmpi.b  #$60,d2         last corumn?
        bne.s   ClrEoL1         branch if not
        rts
****************************************************************
*       get character address
*       input: nothing
*       output: a0.l=chr point
* destroy: d2.l
*       calls: nothing
****************************************************************
ChrAdd
        moveq   #0,d2           init d2
        move.b  CurY(a2),d2     get row(Y)
        ror.w   #5,d2           Y*$800(0->0,31->$f800)
        add.b   CurX(a2),d2     add corumn(X)
        movea.l d2,a0           point to chr offset
        adda.l  #T3U,a0         point to chr address
        rts
****************************************************************
*               character reverce
*       input: a0=chr pointer
*       output: nothing
*       calls: nothing
****************************************************************
ChrRev
        not.b   (a0)            reverse
        not.b   $80(a0)
        not.b   $100(a0)
        not.b   $180(a0)
        not.b   $200(a0)
        not.b   $280(a0)
        not.b   $300(a0)
        not.b   $380(a0)
        not.b   $400(a0)
        not.b   $480(a0)
        not.b   $500(a0)
        not.b   $580(a0)
        not.b   $600(a0)
        not.b   $680(a0)
        not.b   $700(a0)
        not.b   $780(a0)
        rts
****************************************************************
*               brink cursor
*               input -> output
*case1  00000001        11111111 normal(rev:0->1) goto case2
*case2  11111111        00000001 normal(rev:1->0) goto case1
*brk was masked by Write routine
*case3  00000000        00000000 no rev,goto case1
*case4  11111110        00000000 do rev,goto case1
*
* NOTE: The bit 0 of case3 and 4 was cleared by "Write"'s
* head-commands(bsr Brink) only! And certainly that result val-
* ue Brk(a2)=$00 will be changed to normal value Brk(a2)=$01 by
* "Write"'s tail-commands(bsr Brink).
*
* destroy: nothing
*       calls: ChrAdd,ChrRev
****************************************************************
Brink
        move    sr,-(a7)
        move    IRQMask(a2),sr
        tst.b   Brk(a2)         check N flag
        btst.b  #0,Brk(a2)      is brink mask?(N=no change!)
        bne.s   Brink1          branch if not
        bpl.s   Brink9          branch if N(valid!)=0
        clr.b   Brk(a2)         clear mask & brink flag bits
Brink1
        neg.b   Brk(a2)         switch bit 7 of brink flag
        bsr     ChrAdd          get cursor position
        bsr.s   ChrRev          reverse cursor
Brink9
        move    (a7)+,sr
        rts
****************************************************************
*               line delet
*       input: nothing
*       output: nothing
* destroy: d0.l,d1.b
*       calls: Rowcpy
****************************************************************
LinDel
        moveq   #0,d0           initialize
        move.b  CurY(a2),d0     get # of row
        move.b  d0,d1           save
        addi.b  #1,d0           get # of source row
        lsl.w   #8,d0           upper byte is row # of source
        move.b  d1,d0           restore # of destination row
LinDel1
        bsr.s   Rowcpy          copy sce to des whole
        addi.w  #$0101,d0       next row
        cmpi.b  #$20,d0         end of row?
        bne.s   LinDel1         branch if not
        clr.b   CurX(a2)        restore X
        rts
****************************************************************
*               line insert
*       input: nothing
*       output: CurX
* destroy: d0.l
*       calls: Rowcpy
****************************************************************
LinIn
        move.l  #$1e1f,d0       row # of sce($1e) & des($1f)
        bra.s   LinIn2
LinIn1
        bsr.s   Rowcpy          row copy
        subi.w  #$0101,d0       adjust row # of sce & des
LinIn2
        cmp.b   CurY(a2),d0     last row?
        bne.s   LinIn1          branch if not
        move.w  #$2000,d0       set row # of source
        move.b  CurY(a2),d0     merge row # of destination
        bsr.s   Rowcpy          row copy
        clr.b   CurX(a2)        restore X
        rts
****************************************************************
*               clear text(T3:$e60000-$e6ffff)
*       input: nothing
*       output: nothing
* destroy: d0.w,d2.l
*       calls: Rowcpy
****************************************************************
ClrTxt
        move.l  #$2000,d2       set row # of sce & des
ClrTxt1
        move.w  d2,d0           move parameter
        bsr.s   Rowcpy          row copy
        addi.b  #1,d2           next row
        cmpi.b  #$20,d2         last row?
        bne.s   ClrTxt1         branch if not
        clr.b   CurX(a2)        restore X
        clr.b   CurY(a2)        restore Y
        rts
****************************************************************
*                                       schroll
*       input: nothing
*       output: nothing
* destroy: d0.l
*       calls: Rowcpy
****************************************************************
Schroll
        move.l  #$0100,d0       set row # of sce & des
Schroll1
        bsr.s   Rowcpy          row copy
```

```
        addi.w  #$0101,d0       next row
        cmpi.b  #$20,d0         last row?
        bne.s   Schroll1        branch if not
        rts
*****************************************************************
*               row copy
*       input: d0.w=(sce row Num).b(dst row Num).b
*       output: nothing
* destroy: d1.l,d2.w
*       calls: Rstcpy
*****************************************************************
Rowcpy
        move.w  d0,d2           save
        lsl.w   #2,d0           a row is four raster(=4dot)
        moveq   #3,d1           set loop count
Rowcpy1
        bsr.s   Rstcpy          raster(=4 dot line) copy
        addi.w  #$0101,d0       next raster
        dbra    d1,Rowcpy1      count & branch if continue
        move.w  d2,d0           restore for next call
        rts
*****************************************************************
*               Raster copy
*       input: d0.w=(sce lust Num).b(dst lust Num).b
*       output: nothing
* destroy: nothing
*       calls: noting
*****************************************************************
Rstcpy
        movea.l #CRTC,a3        point to CRTC
        move.w  #$8,C.TSel(a3)  set # of txt plane
        move    sr,-(a7)        save
        ori.w   #$0700,sr       mask
Rstcpy1
        btst.b  #7,$e88001      H-SYNC?
        beq.s   Rstcpy1         branch if high
        move.w  d0,C.Ras(a3)    set raster address of sce & des
        move.w  #8,C.Mode(a3)   raster copy
Rstcpy2
        btst.b  #7,$e88001      H-SYNC?
        beq.s   Rstcpy2         branch if high
        clr.w   C.Mode(a3)      stop raster copy
        move.w  (a7)+,sr        restore
        rts

*****************************************************************
*               Getsta/Putsta
*       Get/Put Acia Status
* Passed: (d0.w)=Status Code
*               (a1)=Path Descriptor
*               (a2)=Static Storage address
* returns: depends on status code
*****************************************************************
GetStat
        cmpi.w  #SS_Ready,d0    Ready status?
        bne.s   GetSta10        ..no
        movea.l PD_RGS(a1),a0   get caller's register stack
        clr.w   R$d1(a0)        sweep reg
        moveq   #0,d0
        move.b  Count(a2),d0    get # of rest
        move.w  d0,R$d1+2(a0)   ret input char count to caller
        beq     ErrNtRdy        ..No data; ret not ready error
        rts                     (Carry clear)
GetSta10
        cmpi.b  #SS_EOF,d0      End of file?
        beq.s   GetSta99        ..yes; return (Carry clear)
Unknown
        move.w  #E$UnkSvc,d1    Unknown service code
        ori     #Carry,ccr      return Carry set
GetSta99
        rts
PutStat
        cmpi.w  #SS_SSig,d0     signal process when ready?
        bne.s   PutSta_A        ..No
        tst.w   SigPrc(a2)      somebody already waiting?
        bne     ErrNtRdy        ..Yes; error
        move.w  PD_CPR(a1),d0   get caller's process ID
        movea.l PD_RGS(a1),a0   get caller's register ptr
        move.w  R$d2+2(a0),d1   get signal code
        move    sr,-(a7)        save IRQ status
        move    IRQMask(a2),sr  disable IRQs
        tst.w   Count(a2)       any Data available?
        bne.s   PutSta10        yes, signal Data ready
        move.w  d0,SigPrc(a2)   save process ID
        move.w  d1,SigPrc+2(a2) save the desired signal code
        move     (a7)+,sr       unmask IRQs
        moveq   #0,d1           clear carry
        rts
PutSta10
        move    (a7)+,sr        restore IRQ status
        bra     SendSig         send the signal
PutSta_A
        cmpi.w  #SS_Relea,d0    Release Device?
        bne.s   PutSta_B        bra if not
        move.w  PD_CPR(a1),d2   get current process ID
        lea     SigPrc(a2),a3   test SigPrc
ClearSig
        cmp.w   (a3),d2         does it concern this process?
        bne.s   ClrSig20        ..no; just return
        clr.w   (a3)            no more signals for him
ClrSig20
        moveq   #0,d1
        rts
PutSta_B
        bra.s   Unknown
*****************************************************************
*               Subroutine TrmNat
*       Terminate Acia processing
* Passed: (a1) device descriptor pointer
*               (a2)=static storage
*                (a4)=current process descriptor ptr
* Returns: none
*****************************************************************
TRMN00
        move.w  V_BUSY(a2),V_WAKE(a2)   arrange wake up signal
        move    (a7)+,sr                restore IRQs
TRMN01
        moveq   #0,d0                   sleep forever
        os9     F$Sleep
        tst.w   V_WAKE(a2)              wake signal?
        bne.s   TRMN01                  branch if not
TrmNat
        move.w  P$ID(a4),d0             get this process ID
        move.w  d0,V_BUSY(a2)           set active process ID
        move.w  d0,V_LPRC(a2)           set last act procs ID
        move    sr,-(a7)                save current IRQ status
        move    IRQMask(a2),sr          mask IRQs
        tst.b   Count(a2)               any data?
        bne.s   TRMN00                  sleep if there is
        movea.l #MFP,a3
        andi.b  #$e1,M.IERA(a3)         disable IRQ of USART
        andi.b  #$e1,M.IMRA(a3)         mask IRQ
        andi.b  #$fe,M.RSR(a3)          disable Receiver
        andi.b  #$fe,M.TSR(a3)          disable Transemitter
        move    (a7)+,sr                restore IRQ masks
        move.b  M$Vector(a1),d0         get vector #
        suba.l  a0,a0
        OS9     F$IRQ                   remove USART
        rts

*****************************************************************
*               ACIRQ
* Process IRQ (input or output) from Acia
* Passed: (a2)=Static Storage addr
*               (a3)=Port address
* Returns: cc=carry set if false IRQ
*****************************************************************
IRQSvc
        move    sr,-(a7)          save
        move    IRQMask(a2),sr    mask IRQ
        movea.l #MFP,a3           point MFP
```

〔リスト 8〕vterm デバイス・ドライバ・モジュール ④

```
        move.b  M.RSR(a3),d0        get receive status
        andi.b  #%0111000,d0        error?
        bne.s   RERR                branch if so
        moveq   #0,d2
        move.b  M.UDR(a3),d2        read data(key code)
        move.l  d2,d1
        cmpi.b  #$ff,d2             claim to set LED?
        beq.s   LEDcmd              branch if so
        andi.b  #$7f,d1             get key code
        beq.s   RERR                btanch if error(0)
        cmpi.b  #$6c,d1             key code =< $6c?
        bls.s   Code                branch if so
        cmpi.b  #$70,d1             key code > $70?
        bcs.s   RERR                branch if so
        cmpi.b  #$77,d1             key code =< $77?
        bls.s   Code                branch if so
RERR
        move.b  d0,V_ERR(a2)        set error code(receive code)
        bclr.b  #0,M.RSR(a3)        USART reset
        bset.b  #0,M.RSR(a3)        USART restart
IRQEnd
        move    (a7)+,sr            restore
        moveq   #0,d1
        rts
LEDcmd
        btst.b  #1,Cndition(a2) caps lock flag?
        beq.s   LEDcmd0             branch if off
        bclr.b  #3,LEDBit(a2)       caps LED ON
        bra.s   LEDcmd1
LEDcmd0
        bset.b          #3,LEDBit(a2)   caps LED OFF
LEDcmd1
        move.b  #$05,M.TSR(a3)  enable trasmitter
LEDcmd2
        btst.b  #7,M.TSR(a3)        ready to write?
        beq.s   LEDcmd2             branch if not
        move.b  LEDBit(a2),M.UDR(a3) write
LEDcmd3
        btst.b  #7,M.TSR(a3)        transmit buffer empty?
        beq.s   LEDcmd3             branch if not
        move.b  #$04,M.TSR(a3)  disable transmitter
        bra.s   IRQEnd
Code
        cmpi.b  #$70,d1             SHIFT?
        beq.s   SHIFT
        cmpi.b  #$71,d1             CTRL?
        beq.s   CTRL
        cmpi.b  #$5d,d1             CAPS?
        beq.s   CAPS
        btst.l  #7,d2               key off?
        beq.s   Code1               branch if not
        bra.s   IRQEnd
SHIFT
        btst    #7,d2               shift on?
        beq.s   SHIFT1              branch if so
        bclr.b  #2,Cndition(a2) reset flag of shift
        bra.s   IRQEnd
SHIFT1
        bset.b  #2,Cndition(a2) set flag of shift
        bra.s   IRQEnd
CTRL
        btst.l  #7,d2               ctrl on?
        beq.s   CTRL1               branch if so
        bclr.b  #3,Cndition(a2) reset flag of ctrl
        bra.s   IRQEnd
CTRL1
        bset.b  #3,Cndition(a2) set flag of ctrl
        bra     IRQEnd
CAPS
        btst.l  #7,d2               caps on?
        beq.s   CAPS1               branch if so
        bclr.b  #0,Cndition(a2) reset flag of caps
        bra     IRQEnd
CAPS1
        btst.b  #0,Cndition(a2) caps flag?
        beq.s   CAPS2               branch if first on-key
        bra     IRQEnd              exit if repeat
CAPS2
        bset.b  #0,Cndition(a2) set flag of caps
        bchg.b  #1,Cndition(a2) swich of caps lock
        bra     LEDcmd
Code1
        cmpi.b  #$FF,Count(a2)      buffer full?
        beq     IRQEnd              ...yes;ret with no operation!
        moveq   #0,d2
        btst.b  #3,Cndition(a2) Ctrl code?
        beq     Code2               branch if not
Ctrl
        lea     TableCTRL(pc),a0 point to table of CTRL
        moveq   #0,d2
        move.b  Next(a2),d2         get pointer of loop buffer
        sub.b   Count(a2),d2        adjust pointer
        lea     Buff(a2),a1         point to buffer
        move.b  0(a0,d1),d0         get DATA
        cmp.b   V_INTR(a2),d0       is it interrupt chr?
        beq.s   Ctrl_c              branch if so
        cmp.b   V_QUIT(a2),d0       is it quit chr?
        beq.s   Ctrl_e              branch if so
        cmp.b   V_PCHR(a2),d0       is it pause chr?
        beq.s   Ctrl_w              branch if so
Ctrl1
        move.b  d0,0(a1,d2)         put buffer
        addi.b  #1,Count(a2)        adjust counter
        bra.s   WakeUp
Ctrl_w
        tst.l   V_DEV2(a2)          any echo device?
        beq.s   Ctrl1               branch if not
        movea.l V_DEV2(a2),a0       get echo device static ptr
        move.b  d0,V_PAUS(a0)       request pause
        bra.s   Ctrl1
Ctrl_c
        moveq   #S$Intrpt,d1        set interrupt signal
        bra.s   Ctrl_e1
Ctrl_e
        moveq   #S$Abort,d1         set abort signal
Ctrl_e1
        move.b  d0,-(a7)            save input chr
        move.w  V_LPRC(a2),d0       get last process ID
        bsr.s   Wake10              send error signal
        move.b  (a7)+,d0            restore input chr
        bra.s   Ctrl1
Code2
        btst.b  #1,Cndition(a2) Caps ?
        beq.s   Code3               branch if not
        btst.b  #2,Cndition(a2) Shift ?
        beq.s   Code_C              branch if not
        lea     TableCS(pc),a0  point to table of CAPS & SHIFT
        bra.s   CodeEnd
Code_C
        lea     TableC(pc),a0       point to table of CAPS
        bra.s   CodeEnd
Code3
        btst.b  #2,Cndition(a2) Shift ?
        beq.s   Code_N              branch if not
        lea     TableS(pc),a0       point to table of SHIFT
        bra.s   CodeEnd
Code_N
        lea     TableN(pc),a0       point to table of NORMAL
CodeEnd
        moveq   #0,d2
        move.b  Next(a2),d2         get pointer of loop buffer
        sub.b   Count(a2),d2        adjust pointer
        lea     Buff(a2),a1         point to buffer
        move.b  0(a0,d1),0(a1,d2)           put DATA
        addi.b  #1,Count(a2)        adjust counter
WakeUp
        move.w  SigPrc(a2),d0       any process to notify?
        beq.s   WakeUp05            branch if not
        move.w  SigPrc+2(a2),d1 get signal code
```

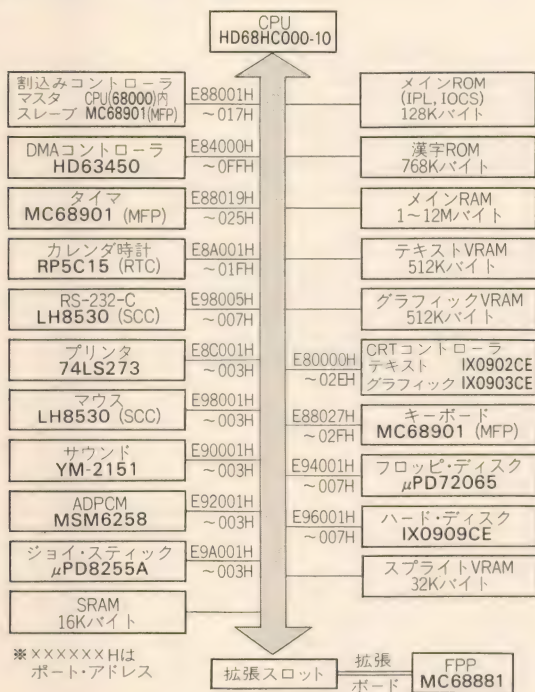
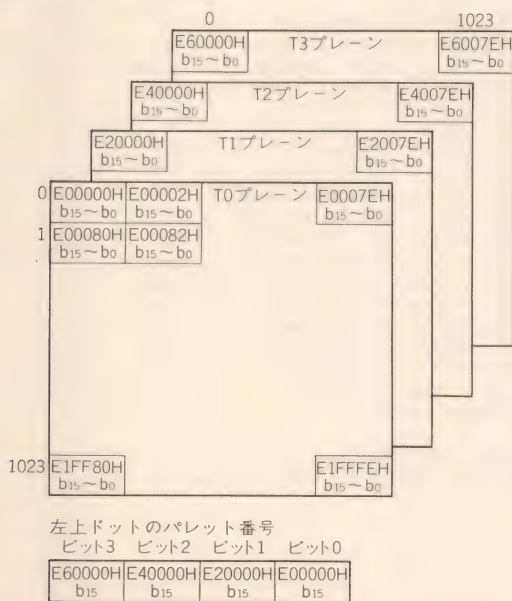
```
        clr.w   SigPrc(a2)       disable signal code
        bra.s   SendSig
WakeUp05
        move    (a7)+,sr
        moveq   #S$Wake,d1       Wake up signal
        move.w  V_WAKE(a2),d0    Owner waiting?
Wake10
        beq.s   Wake90           ..no; return
        clr.w   V_WAKE(a2)
SendSig
        move.l  a0,-(a7)
        movea.l D_SysDis(a6),a0 get system dispatch ptr
*get ptr to send routine
        movea.l F$Send+F$Send+F$Send+F$Send(a0),a0
        jsr     (a0)             send signal
        movea.l (a7)+,a0
Wake90
        moveq   #0,d1
        rts

***************************************************************
*             Tables(each size = 128 byts)
* TableN:Normal keys
* TableS:Shift
* TableC:Caps
* TableCS:Caps & Shift
* TableCTRL:Control
***************************************************************
TableN
 dc.b 0,$1b,"1234567890-^",$5c,8
 dc.b 9,"qwertyuiop@[",$d,"as"
 dc.b "dfghjkl;:]","zxcvbn"
 dc.b "m",$2C,"./",0,$20,$b,$7f,0,0,0,$1d,$1e,$1c,$1f,$c
 dc.b "/*-789+456=123",$d,"0"
 dc.b $2c,".",0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
TableS
 dc.b 0,$1b,"!",$22,"#$%&'()",0,"=¯|",8
 dc.b 9,"QWERTYUIOP@{",$d,"AS"
 dc.b "DFGHJKL+*}","ZXCVBN"
 dc.b "M<>?_",$20,$b,$7f,0,0,0,$1d,$1e,$1c,$1f,$c
 dc.b "/*-789+456=123",$d,"0"
 dc.b $2c,".",0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
TableC
 dc.b 0,$1b,"1234567890-^",$5c,8
 dc.b 9,"QWERTYUIOP@[",$d,"AS"
 dc.b "DFGHJKL;:]","ZXCVBN"
 dc.b "M",$2c,"./",0,$20,$b,$7f,0,0,0,$1d,$1e,$1c,$1f,$c
 dc.b "/*-789+456=123",$d,"0"
 dc.b $2c,".",0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
TableCS
 dc.b 0,$1b,"!",$22,"#$%&'()",0,"=¯|",8
 dc.b 9,"qwertyuiop@{",$d,"as"
 dc.b "dfghjkl+*}","zxcvbn"
 dc.b "m<>?_",$20,$b,$7f,0,0,0,$1d,$1e,$1c,$1f,$c
 dc.b "/*-789+456=123",$d,"0"
 dc.b $2c,".",0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
TableCTRL
 dc.b 0,$1b,0,0,0,0,0,0,0,0,0,0,0,$1e,$1c,8
 dc.b $9,$11,$17,$5,$12,$14,$19,$15,$9,$f,$10,0,$1b,$d,$1,$13
 dc.b $4,$6,$7,$8,$a,$b,$c,0,0,$1d,$1a,$18,$3,$16,$2,$e
 dc.b $d,0,0,0,$1f,$20,$b,$7f,0,0,0,$1d,$1e,$1c,$1f,$c
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,$d,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 dc.b 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

 ends
```

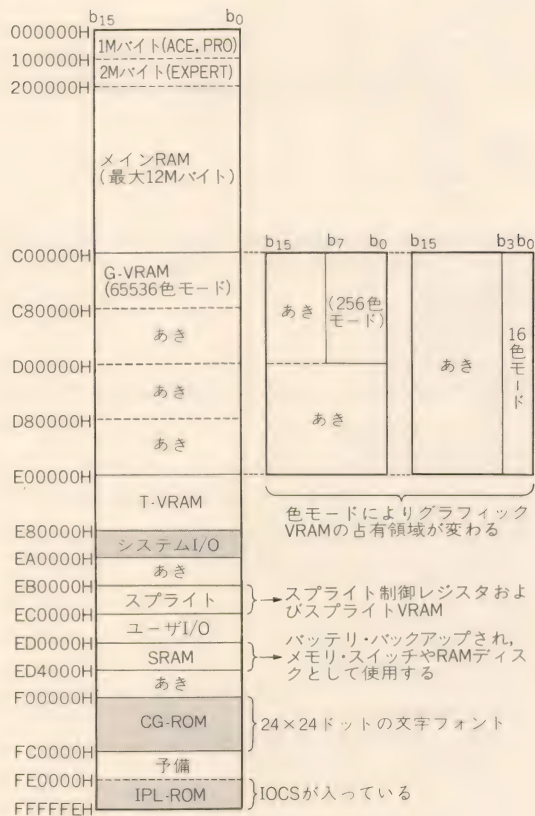
〔リスト 9〕term デバイス・デスクリプタ・モジュール

```
  nam Term
  ttl X68000 Term device descriptor module
  use <defsfile>

 Edition  equ 4 current edition number
 TypeLang set (Devic<<8)+0
 Attr_Rev set (ReEnt<<8)+0
  psect ScfDesc,TypeLang,Attr_Rev,Edition,0,0

 Port     equ $e88000 Port address
 Vector   equ $4c autovector number
 IRQLevel equ $6 hardware interrupt level
 Priority equ $1 polling priority
 Parity   equ $0 parity, stop bits
 BaudRate equ $0 baud rate
 EchoNam  equ bname echo device descriptor (self)
 Mode     set ISize_+Updat_ default device mode capabilities

  dc.l Port port address
  dc.b Vector auto-vector trap assignment
  dc.b IRQLevel IRQ hardware interrupt level
  dc.b Priority irq polling priority
  dc.b Mode Device mode capabilities
  dc.w FileMgr file manager name offset
  dc.w DevDrv device driver name offset
  dc.w 0 DevCon (reserved)
  dc.w 0,0,0,0 reserved
  dc.w OptSiz option byte count

* Default Parameters
Options
*                                              default
*       name     function                      value
*       -------- --------------------          -------
 dc.b DT_SCF   device type                     SCF
 dc.b upclock  upcase lock                     OFF
 dc.b bsb      backspace=BS,SP,BS              ON
 dc.b linedel  line del/bsp line               OFF
 dc.b autoecho full duplex                     ON
 dc.b autolf   auto line feed                  ON
 dc.b eolnulls null count                      0
 dc.b pagpause end of page pause               OFF
 dc.b 32 pagsize  lines per page               24
 dc.b C$Bsp    backspace char                  ^H
 dc.b C$Del    delete line char                ^X
 dc.b C$CR     end of record char              <return>
 dc.b C$EOF    end of file char                ESC
 dc.b C$Rprt   reprint line char               ^D
 dc.b C$Rpet   dup last line char              ^A
 dc.b C$Paus   pause char                      ^W
 dc.b C$Intr   Keyboard Interrupt char         ^C
 dc.b C$Quit   Keyboard Quit char              ^E
 dc.b C$Bsp    backspace echo char             ^H
 dc.b C$Bell   line overflow char              ^G
 dc.b Parity   stop bits and parity            none
 dc.b BaudRate bits/char and baud rate         none
 dc.w EchoNam  offset of echo device           none
 dc.b C$XOn    Transmit Enable char            ^Q
 dc.b C$XOff   Transmit Disable char           ^S
 dc.b C$Tab    tab character                   ^I
 dc.b tabsize  tab column size                 4
OptSiz equ *-Options

FileMgr dc.b "Scf",0  file manager
DevDrv dc.b "vterm",0 driver module name

 ends
```

X68000 マシン・システムの概要

（黒田 宣孝）

● X68000 のシステム構成図

●テキスト画面構成

● テキスト画面の色は，パレットにより65,536色中の16色を選択できる

●メモリ・マップ

●テキスト画面モード

CRT モード	解像度(水平同期)	表示サイズ	文字数(8×16 フォント)
0，4，8，12	高解像度(31 kHz)	512×512	64 文字×32 行
1，5，9，13	低解像度(15 kHz) オーバスキャン	512×512	64 文字×32 行
2，6，10，14	高解像度(31 kHz)	256×256	32 文字×16 行
3，7，11，15	低解像度(15 kHz) オーバスキャン	256×256	32 文字×16 行
16	高解像度(31 kHz)	768×512	96 文字×32 行
17	中解像度(24 kHz)	1024×400	128 文字×26 行
18	中解像度(24 kHz)	1024×800	128 文字×53 行

● CRT モード：IOCS により設定できる

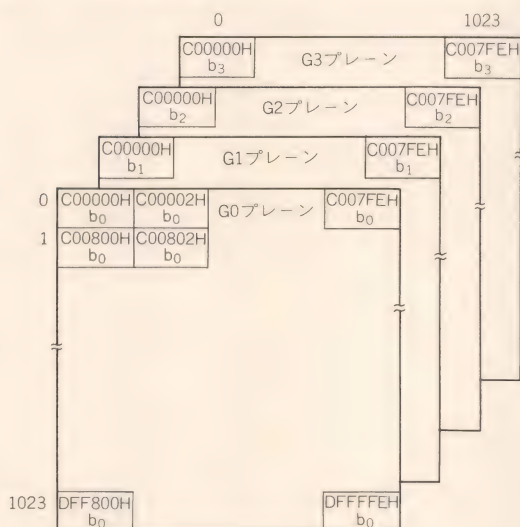
● CG-ROM アドレス・マップ

アドレス	フォント構成	文字の種類
F00000H	16×16	非漢字 752 文字
F05E00H	16×16	第 1 水準漢字 3008 文字
F1D600H	16×16	第 2 水準漢字 3478 文字
F388C0H		あき
F3A000H	8×8	ASCII 文字 256 文字
F3A800H	8×16	ASCII 文字 256 文字
F3B800H	12×12	ASCII 文字 256 文字
F3D000H	12×24	ASCII 文字 256 文字
F40000H	24×24	非漢字 752 文字
F4D380H	24×24	第 1 水準漢字 3008 文字
F82180H	24×24	第 2 水準漢字 3478 文字
FBF3B0H		あき

● 文字の表示は，ソフトウェアにより CG-ROM のフォント・パターンをテキスト VRAM に転送することで行う

● 割込み一覧

レベル	デバイス	割込み要因	
高 7	NMI	NMI スイッチ	
6	MFP	優先順位	周辺 LSI MFP による
		高 15	CRTC からの H-SYNC 信号
		14	CRTC からの IRQ 信号
		13	CRTC からの V-DISP 信号
		12	キー・データの受信割込み
		11	キー・データの受信エラー
		10	キー・データの送信割込み
		9	キー・データの送信エラー
		8	USART シリアル・クロック
		7	未使用
		6	CRTC の V-DISP 信号の状態検出
		5	8 ビット汎用タイマ 1 (クロック 4 MHz)
		4	8 ビット汎用タイマ 2 (クロック 4 MHz)
		3	FM 音源
		2	電源スイッチによる ON/OFF 検出
		1	EXPWON 信号による ON/OFF 検出
		低 0	ALARM 信号による ON/OFF 検出
5	SCC	RS-232-C，マウス・データ受信	
4	—	拡張 I/O スロット	
3	DMAC	DMA 転送終了など	
2	—	拡張 I/O スロット	
低 1	ディスク/プリンタ	優先順位	カスタム LSI IX0905CE による
		高 3	FDC(フロッピ・ディスク・コントローラ)
		2	FDD(フロッピ・ディスク・ドライブ)
		1	ハード・ディスク
		低 0	プリンタ BUSY

● グラフィック画面構成(1024×1024 ドット 16 色モード)

左上ドットのパレット番号

ビット3	ビット2	ビット1	ビット0
C00000H b_3	C00000H b_2	C00000H b_1	C00000H b_0

● グラフィック画面モード

CRT モード	解像度(水平同期)	実画面	表示サイズ	色数	面数
0	高解像度(31 kHz)	1024×1024	512×512	16	1
1	低解像度(15 kHz) オーバスキャン	1024×1024	512×512	16	1
2	高解像度(31 kHz)	1024×1024	256×256	16	1
3	低解像度(15 kHz) オーバスキャン	1024×1024	256×256	16	1
4	高解像度(31 kHz)	512×512	512×512	16	4
5	低解像度(15 kHz) オーバスキャン	512×512	512×512	16	4
6	高解像度(31 kHz)	512×512	256×256	16	4
7	低解像度(15 kHz) オーバスキャン	512×512	256×256	16	4
8	高解像度(31 kHz)	512×512	512×512	256	2
9	低解像度(15 kHz) オーバスキャン	512×512	512×512	256	2
10	高解像度(31 kHz)	512×512	256×256	256	2
11	低解像度(15 kHz) オーバスキャン	512×512	256×256	256	2
12	高解像度(31 kHz)	512×512	512×512	65536	1
13	低解像度(15 kHz) オーバスキャン	512×512	512×512	65536	1
14	高解像度(31 kHz)	512×512	256×256	65536	1
15	低解像度(15 kHz) オーバスキャン	512×512	256×256	65536	1
16	高解像度(31 kHz)	1024×1024	768×512	16	1
17	中解像度(24 kHz)	1024×1024	1024×400	16	1
18	中解像度(24 kHz)	1024×1024	1024×800	16	1

● IOCS 一覧

番号	機　　能
$00	キー・データの読出し
$01	キー・データ・バッファ状態のセンス
$02	シフト・キー状態のセンス
$03	キー関係の初期化
$04	キー入力状態のセンス
$0C	TV コントロール
$0D	LED キーのモード指定
$0E	テキスト/グラフィック画面の使用状態
$0F	外字パターンの登録
$10	CRT モードの指定
$11	コントラストの指定
$12	HSV から RGB への色変換
$13	テキストのパレット指定
$15	テキストのカラー指定
$16	漢字パターンのアドレス取得
$17	VRAM からバッファへのバイト転送
$18	バッファから VRAM へのバイト転送
$19	漢字パターンのメモリへの転送
$1A	テキスト画面パターンのメモリへの転送
$1B	メモリ・パターンのテキスト画面への転送
$1C	$1B の機能+クリッピング処理
$1D	テキスト/グラフィック画面の表示座標
$1E	カーソル ON
$1F	カーソル OFF
$20	1文字表示
$21	文字列表示
$22	文字属性の設定
$23	カーソル位置の設定/取得
$24	カーソル1行下/スクロール・アップ
$25	カーソル1行上/スクロール・ダウン
$26	カーソル *N* 行上(スクロールなし)
$27	カーソル *N* 行下(スクロールなし)
$28	カーソル *N* 桁右(スクロールなし)
$29	カーソル *N* 桁左(スクロールなし)
$2A	画面クリア
$2B	カーソル行クリア
$2C	カーソル位置 *N* 行挿入
$2D	カーソル位置 *N* 行削除
$2E	表示範囲の指定
$2F	ファンクション行の文字列表示
$30	RS-232-C の設定
$31	RS-232-C 受信バッファのデータ数
$32	RS-232-C 受信データの取得
$33	RS-232-C 受信データのセンス
$34	RS-232-C 送信状態のセンス
$35	RS-232-C データの送信
$3B	ジョイ・スティック・データの読出し
$3C	プリンタ・ポートの初期化
$3D	プリンタ出力状態のセンス
$3E	プリンタ出力(データ直接)
$3F	プリンタ出力(シフト JIS 漢字処理)
$40	フロッピ/ハード・ディスクのシーク
$41	ディスク・データの比較チェック
$42	フロッピ・ディスクの診断読出し
$43	ディスク関係の初期化
$44	ディスク・ステータスの調査
$45	フロッピ/ハード・ディスクの書込み
$46	フロッピ/ハード・ディスクの読出し
$47	トラック0へのシーク
$48	ハード・ディスク代替トラックの設定
$49	フロッピ・ディスク破損データ書込み
$4A	フロッピ・ディスク ID 情報の読出し
$4B	ハード・ディスクにバッド・トラックの設定
$4C	フロッピ・ディスク破損データ読出し
$4D	ディスクの物理フォーマット
$4E	フロッピ・ドライブの状態センス
$4F	ディスク・イジェクト
$50	日付データの BCD 変換
$51	カレンダ時計への日付設定
$52	時刻データの BCD 変換
$53	カレンダ時計への時刻設定
$54	カレンダ時計からの日付読出し
$55	日付データのバイナリ変換
$56	カレンダ時計からの時刻読出し
$57	時刻データのバイナリ変換
$58	日付文字列のバイナリ変換
$59	時刻文字列のバイナリ変換
$5A	日付データの文字列変換
$5B	時刻データの文字列変換
$5C	曜日データの文字列変換
$5D	アラーム機能の禁止/許可
$5E	アラーム機能のデータ設定
$5F	アラーム機能のデータ取得
$60	ADPCM のデータ出力
$61	ADPCM のデータ入力
$62	ADPCM のデータ出力
$63	ADPCM のデータ入力
$64	ADPCM のデータ出力
$65	ADPCM のデータ入力
$66	ADPCM 実行モードのセンス
$67	ADPCM の終了/中止/再開
$68	FM 音源へのデータ書込み
$69	FM 音源のステータス読出し
$6A	FM 音源による割込みの設定
$6B	MFP のタイマ2による割込みの設定
$6C	CRTC の V-DISP による割込み
$6D	CRTC の IRQ 信号による割込み
$6E	CRTC の H-SYNC による割込み
$6F	プリンタによる割込みの設定
$70	マウスの初期化
$71	マウス・カーソルの表示
$72	マウス・カーソルの消去
$73	マウス・カーソル表示モードのセンス
$74	マウスの移動量/ボタン状態のセンス
$75	マウス・カーソル座標のセンス
$76	マウス・カーソル座標の設定
$77	マウス・カーソル移動範囲の設定
$78	マウス・ボタンを離すまでの時間調査
$79	マウス・ボタンを押すまでの時間調査
$7A	マウス・カーソルのパターン定義
$7B	マウス・カーソルの選択
$7C	マウス・カーソルによるアニメーション
$7D	ソフト・キーボードの制御
$7E	電卓のセンス
$7F	BIOS 起動後の時間
$80	割込みベクタの設定
$81	スーパバイザ・モードへの切替え
$82	データのバイト読出し
$83	データのワード読出し
$84	データのロング・ワード読出し
$85	データのバイト列読出し
$86	データのバイト書込み
$87	データのワード書込み
$88	データのロング・ワード書込み
$89	データのバイト列書込み
$8A	DMA 転送
$8B	DMA 転送(アレイ・チェーン)
$8C	DMA 転送(リンク・アレイ・チェーン)
$8D	DMA 実行モードのセンス
$8E	ブート情報のセンス
$8F	ROM のバージョン/年月日のセンス
$90	グラフィック画面のクリア/表示
$94	グラフィック・パレットの設定
$A0	シフト JIS コードの JIS コード変換
$A1	JIS コードのシフト JIS コード変換
$A2	ANK コードの全角シフト JIS 変換
$A3	ローマ字かな変換
$A4	濁点処理
$A5	半濁点処理
$AE	カーソル ON
$AF	カーソル OFF
$B1	グラフィック画面書込みページの設定
$B2	グラフィック画面表示ページの設定
$B3	グラフィック画面表示位置の設定
$B4	グラフィック画面のウィンドウの設定
$B5	グラフィック画面のクリア
$B6	グラフィック画面のポイント設定
$B7	グラフィック画面のポイント取得
$B8	グラフィック画面の直線描画
$B9	グラフィック画面のボックス描画
$BA	グラフィック画面のボックス塗りつぶし
$BB	グラフィック画面の円/円弧/扇/楕円
$BC	グラフィック画面の塗りつぶし
$BD	グラフィック画面の文字列表示
$BE	グラフィック画面のパターン読出し
$BF	グラフィック画面のパターン書込み
$C0	スプライト面の初期化
$C1	スプライト面の表示 ON
$C2	スプライト面の表示 OFF
$C3	PCG パターンのクリア
$C4	PCG パターンの設定
$C5	PCG パターンの読出し
$C6	スプライト・レジスタの設定
$C7	スプライト・レジスタの読出し
$C8	BG スクロール・レジスタの設定
$C9	BG スクロール・レジスタの読出し
$CA	BG コントロール・レジスタの設定
$CB	BG コントロール・レジスタの読出し
$CC	BG テキストのクリア
$CD	BG テキストの設定
$CE	BG テキストの読出し
$CF	スプライト・パレットの設定
$FD	アボート・フラグのリセット
$FF	アボート

MEMO

デバイス・デスクリプタの構造

(丹下 義郎)

(注1) オフセットは,ヘッダのつぎにくるデスクリプタ本体の先頭からのオフセットを示す.

(a) デバイス・デスクリプタの構造

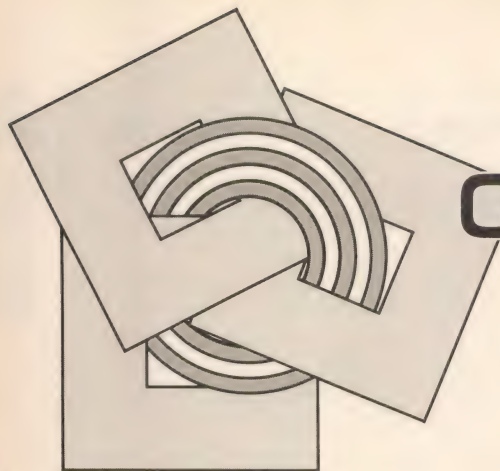
```
    psect    desc,......
Port       equ     $xxxxxxxx
                ~
           dc.w    Port
           dc.b    Vector
           dc.b    IRQLvl
                ~
           dc.w    Manager
           dc.w    Driver
           dc.w    DevCon  (なければ '0' をセット)
           dc.w    0,0,0,0
           dc.w    OptSiz
*
*マネージャに依存するパラメータ領域
*
Option
           dc.b    ..
           dc.w    ..
OptSiz     equ     *-Option
*
*デバイスに依存するパラメータ領域
*          (マネージャは関知しない)
DevCon     dc.w    DevConSiz
                ~
           dc.b    ..
                ~
DevConSiz equ *-DevCon
                ~
Manager dc.b    "RBF",0
Driver     dc.b    "ram",0
    ends
```

(b) コーディング例

オフセット	サイズ	内容
$48	1	デバイス・クラス (0=SCF, 1=RBF, 2=PIPE, 3=SBF, 4=NET)
$49	1	ドライブ番号
$4A	1	ステップ・レート
$4B	1	デバイス・タイプ
$4C	1	デバイス・デンシティ
$4D	1	予約
$4E	2	シリンダ数
$50	1	ヘッド数
$51	1	ベリファイ・フラグ(0=ベリファイを行う)
$52	2	セクタ数/トラック
$54	2	セクタ数/トラック0
$56	2	セグメント割当てサイズ
$58	1	セクタ・インタリーブ・ファクタ
$59	1	DMA転送モード
$5A	1	トラック・ベース・オフセット
$5B	1	セクタ・ベース・オフセット
$5C	2	セクタ・サイズ(バイト)
$5E	2	フォーマット・コントロール (0=フォーマット・イネーブル)
$60	1	リトライ回数(1=リトライをしない)
$61	1	SCSIユニット番号
$62	1	ライト・プリコンペンション開始シリンダ
$63	3	リデュース・ライト・カレント・シリンダ
$66	2	シャット・ダウン時のヘッド固定位置
$68	4	論理セクタ・オフセット
$6C	2	総シリンダ数
$6E	1	SCSIコントローラID番号

(c) オプション領域(RBFタイプの場合)

オフセット	サイズ	内容
$48	1	デバイス・クラス (0=SCF, 1=RBF, 2=PIPE, 3=SBF, 4=NET)
$49	1	文字種 (0=大文字, 小文字可能)
$4A	1	バック・スペースのエコーバック指定
$4B	2	行削除のエコーバック指定
$4D	1	エコーバック (0=エコーバックしない)
$4E	1	自動改行 (0=自動改行しない)
$4F	1	行末でのNULLコード出力数
$50	1	1ページごとの一時停止(0=停止しない)
$51	1	1ページごとの行数
$52	1	行削除のコード
$53	1	レコード終了コード
$54	1	EOFコード
$55	1	カレント・ラインの再表示コード
$56	1	最終入力行の再入力コード
$57	1	出力一時停止コード
$58	1	キーボード割込みシグナル・コード
$59	1	キーボード・アボート・シグナル・コード
$5A	1	バック・スペースでのエコーバック・コード
$5B	1	行オーバフロー時のエコーバック・コード
$5C	1	ストップ・ビット, キャラクタのビット数
$5D	1	ソフトウェア設定のボーレート
$5E	2	出力デバイス名へのオフセット
$60	1	X-ONコード
$61	1	X-OFFコード
$62	1	タブ・コード
$63	1	タブのカラム幅

(d) オプション領域(SCFタイプの場合)

OS-9/68K のもとでの Internet

水野　健

OS-9/68K は, Unix と同様マルチユーザ/マルチタスクOS であるので, Unix と同様のネットワーク・システムを比較的容易に実現できる. 実際マイクロウェアは, Internet をサポートする汎用的なパッケージ ISP, および Ethernet に対応する特定のハードウェアを前提として Internet をサポートする作り付けのパッケージ ESP を発売している. ここでは, これらのソフトウェアにおけるモジュールの階層構造や telnet を経由してリモート login する際のデータの流れなどについて解説する.　　　　(編集部)

1 OS-9/68K のもとでのネットワーク

OS-9/68K のもとで現在サポートされているネットワーク・ソフトウェアには, OS-9/NET と ISP/ESP の 2 種類のものがあります.

(1) OS-9/NET

これは, 複数の OS-9 コンピュータ同士を結合し, このネットワーク内部で対等, ほぼ完全に透明なアクセスを保証するもので, マイクロウェア自身のホスト・マシン群もこの環境のもとで動作しています. ただし, このシステムのもとでは, OS-9 同士のタイトな結合を実現できる一方, 他の OS, たとえば Unix, VMS, MS-DOS マシンなどとの結合には別のネットワークが必要となります. そこでゲートウェイとして開発されたのが, つぎの ISP/ESP パッケージです.

(2) Internet, Ethernet パッケージ

したがって, OS-9 と外の世界との連絡には, ソフトウェア的には Internet, ハードウェア的には Ethernet という選択は, まず自然なものであったということができます.

ここで注意していただきたいのは, Internet とはソフトウェア, またはそのプロトコルの階層関係を包括する表現で, この中に IP, TCP, UDP を含みます. 一方, Ethernet とは, その通路になるハードウェア, およびデータリンク層以下のプロトコル(ハードウェア・プロトコル)を含む, 物理側の表現です. 一般に TCP/IP と Ethernet とを混同する向きもありますが, TCP/IP ソフトウェアがつねにその下位層に Ethernet を使用するとはかぎりません.

2 2 種類の Internet パッケージ

OS-9 のもとでの Internet/Ethernet パッケージには, ESP と ISP の二種類があります.

▶ **ESP** (Ethernet Support Package)

これは, VME バス上で CMC 製 ENP-10 という特定の Ethernet ハードウェアを前提とするパッケージで, 他のハードウェアへの移植は必ずしも考慮されていません. 言い換えると, これは典型的なインテリジェント・ハードウェアを前提として, VME バスで構成される開発用システム(またはターゲット・システム)と中型ホスト・マシンとの結合のために作られ, 出来あい製品(off-the-shelf-use)としてパッケージ化された商品です.

▶ **ISP** (Internet Support Package)

一方, ISP パッケージは, 異なるハードウェアへの移植を仮定したもので, ソフトウェアの論理部と物理部が完全に分離されています. マイクロウェアは OS-9 本体の移植用パッケージをポートパックと呼んでいますが, ISP は Internet/Ethernet のポートパックにあたるものだということができます.

パッケージには, サンプル・デバイス例として, MVME-147 ボード上の Am7990 用のものが付属します. デバイス・ドライバは C 言語で書かれており, このコントローラについては, さまざまなハードウェア例に対応できるよう, ソース・コード上にいくつかのコンパイル・スイッチが入っています.

3 Internet ソフトウェアの階層構造

ESP，ISP には，共通につぎのものが含まれています．

① ftp，telnet コマンドおよびそのバックグラウンド・サーバとなるデーモン・プログラム群

② Unix/bsd 4.3 レベルに等価なソケット・ライブラリ

③ telnet サーバのための仮想端末を作成するソフトウェア・システム

このうち ① ftp，telnet は，当然ながら他の OS のもとで動作する ftp，telnet とまったく同等であり，システム稼働と同時に同一のオペレーションで他の OS との通信ができます．また，OS-9 は Unix と同様にマルチタクス/マルチユーザなので，当然のこととしてバックグラウンド・サーバ群が含まれますが，これらのデーモン・プロセスの起動も基本的に bsd 4.3 と同一であり，要するにオペレータ・レベルでは「あっけない」印象を与えるほどです．したがって，ftp，telnet のオペレーションについては，この記事では説明しません．

② のソケット・ライブラリは，ftp，telnet とそのサーバ・プロセス，その他の既存のソフトウェア，あるいは Unix/bsd のもとで作成されたソフトウェアに，bsd 4.3 とほぼ等価なシステム・サービスを提供するものです．つまり，このライブラリ・インターフェースで，プログラム・レベルの Unix/bsd 互換性が保証されます．このライブラリ内の各関数は，OS-9 のもとで特定の（多くの場合は Ethernet）デバイスにアクセスを行いますが，これについては 4 で説明します．

③ telnet は，いうまでもなく TCP/IP プロトコルを使用して，基本的にはリモート・システム上にログインするためのコマンドです．このコマンドが起動され相手側システムにリクエストが届くと，そのシステム上で（あたかも）通常の端末からログインが行われるかのようにログインが行われます．このとき，概念として要求されるのがログイン端末ですが，ネットワークを介した場合，この端末はそのコンピュータ上に実在しない（ネットワークの向こう側にある）ことになります．

これに対応するにはつぎの二つの方法があります．

(1) telnet によるログイン専用の login，shell を作成する．

(2) 通常の login, shell を動作させ，そのためにソフトウェア的に透明な仮想端末を作成し，「ネットワークの向こう」の端末を実現する．

(1) の方法の例はしばしば見られるものですが，この方法では，いずれ shell がフォークするプロセス群に (2) と同等の仮想端末を用意せざるを得ない，login によるフォークの対象が shell にかぎられる，など解決が繁雑になります．

ESP，ISP で用意されているのは，リモート端末（ネットワークの向こうに接続されている端末）をエミュレートする仮想端末 pseudo-keyboard で，これがバックグラウンド・プロセスによって初期化されたうえで，通常の login, shell に引き継がれます（この間の詳しいデータの流れは 4 で説明する）．

この仮想端末を受けもつものが，pkman（ファイル・マネージャ SCF に相当）と pkdvr（そのデバイス・ドライバ）です．pkman に管理されるこの「ログイン端末」は，アプリケーション・プログラム，つまり login や shell にとって通常の OS-9 端末（SCF デバイス）と変わるところはなく，通常の標準入出力がネットワークを介して，telnet を起動した端末につながれているにすぎません．したがって，shell を通してフォークされるすべてのアプリケーション・プログラムが，リモート・システム上で動作することになります．

4 telnet の上部構造とデータの流れ

そこで，以下 ISP パッケージを例にして，telnet からネットワーク上のデータの流れを簡単に追ってみることにします．

ここで再び注意していただきたいのは，① ここで説明するのは OSI 7 層モデルの第 5 層（セッション層）から上だという点です．つぎに，② 最上位の第 7 層（アプリケーション層）に通常のシステム・ユーティリティ（たとえば shell）が存在する，つまり，これらのアプリケーション・プログラムにとってネットワークは透明に存在する（ネットワークの存在を意識しなくてもよい）のですが，ネットワーク・システムとは本来そのような環境を提供するものであり，第 6 層以下はオペレータ（操作する人）には関係のないものだ，という点．また，③ これ以下のすべてのアクセスはソケット関数群を呼び出すことで実現されるのですが，それ自体はオペレータにとってたんなるブラックボックスにすぎないという点です．

同様に，このソケット関数群は IP プロトコルを実現する論理モジュールであって，これ以下の層についてはソケットから見るとブラックボックスであるにすぎません．OS-9/ISP パッケージでは，ここからさらにデバイス・ドライバとのインターフェースが行われ，ここまでがパッケージの論理部，つまり OSI 第 4 層（トランスポート層）までに該当します．ここから先，第 3 層（ネットワーク層）がデバイス・ドライバと呼ばれ，ISP パッケージでは，はじめて現実のハードウェア

と対面しますが，この説明は 5 で行います．

図1で①～⑫はデータの送受信者，〈A〉～〈E〉はデータ自身の流れを示します．ここでいう④のブラックボックスは，ソケットの背後のデバイス・ドライバを含むハードウェア層だと考えてください．

①のtelnetは，端末②からオペレータが直接起動するコマンドです．このコマンドのオペレーション自体は，どんなOSのもとでも同一で，Unix上でもOS-9上でも(MS-DOSでも)，最終的には相手側loginを介してshellを動作させる，一連の動作をします．ここで①のtelnetと⑥のtelnetdの交換するプロトコルがTCPです．“telnetd”の“d”は“daemon”のことで，このプロセスは通常システム起動時にバックグラウンドで起動され，入力〈B〉を監視しつづけます．

①のtelnetが起動されると，telnetは指定された相手側ステーションにむけてリクエスト〈A〉を発信します．このメッセージは④のブラックボックスを経て相手側の⑥のtelnetdに届けられ(〈B〉)，telnetdは以下の一連の動作を行い，その後は再び他の入力〈B〉の監視に戻ります．

このとき，〈A〉と〈B〉の間で交換されるのはいうまでもなくTCP/IPであり，これはどのOS間でも同一のものです．もしこの互換性が保証されなければ，MS-DOS～Unix～OS-9間の通信はあり得ません．

さて，⑥のtelnetdは，二つの顔をもっています．一つは，異なるOSのもとで起動されたかもしれない(たとえば，ネットワーク上に混在するMS-DOS，Unixの)telnetからの入力メッセージ〈B〉を受けとり，標準プロトコルにしたがって通信を行い，相手側にloginの起動開始を伝えること．第2の顔は，それにともなって特定のOS(ここではOS-9)環境のもとで，相手側のためにloginをフォークすることです．

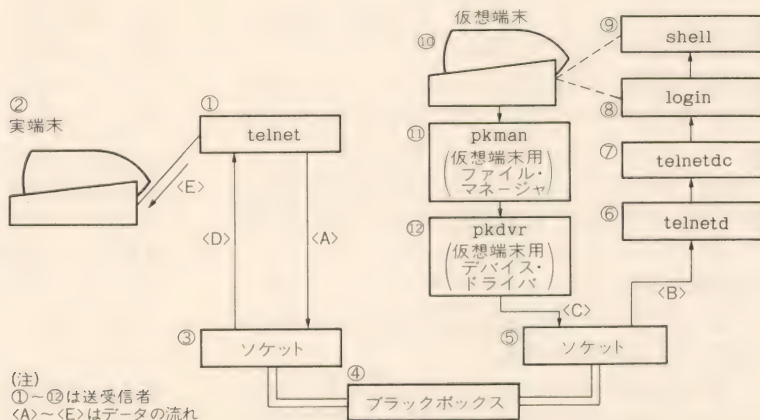
この第2の点は，新プロセス起動時のオーバヘッドを避けるために，もう一つのデーモン・プロセスtelnetdc(⑦)にまかされます．そしてtelnetd自身は，telnetdcのフォーク後，ただちに入力〈B〉の監視に戻ります．

⑦のtelnetdcは，純然たるOS-9プログラムであり，つぎの動作をします．

(1) telnetdからソケットのIDを受けとり，それに対応する仮想端末(⑩)をこのシステム上に生成する．
(2) 生成された仮想端末を標準入出力としてもつloginプロセスをフォークする(loginはやがてshellにチェーンする)．
(3) 子プロセス(ここではshell)が最終的にlogoutするまで待機し，終了後，telnetdにそれを通知すると同時に，仮想端末をクローズする．

ここで生成された仮想端末は，loginからshellへ，さらにshellからフォークされるすべての子プロセスに引き継がれていき，それらのプロセスは標準入出力として「ネットワークの向こうの」端末(②)と対話することになります．この仮想端末(⑩)を介したネットワーク端末(②)への通信について，以下に述べます．

3 でも述べましたが，この仮想端末は，あたかも通常のSCFデバイスのように振るまいます．loginやshellにとってこの「端末」は純然たるSCFデバイスであり，通常の端末I/O操作でSCFと同等の動作が行われます．この管理を行っているのがpkman，そしてI/O呼出しにともなってソケット・コールを行い，相手側(telnet端末②側)システムとの交信を行うのがデバイス・ドライバpkdvrです．このデータの流れ(〈C〉)は，リクエスタであるtelnet(①)のためのものであり，したがってTCP/IP，つまりtelnetプロトコルで行われます．この転送は図でブラックボックスを経由してtelnetに戻され(〈D〉)，実端末(②)に渡されます(〈E〉)．なお，〈C〉，〈D〉，〈E〉の流れは図では片方向に描いていますが，実際には文字の入出力に対応する双方向です．

〔図1〕
ISPパッケージの下でのデータの流れ

5 ソケットの下部構造とインテリジェント・ハードウェアの問題

④で「ブラックボックス」として取り扱った部分は，内容的には物理部(具体的なネットワーク・ハードウェアとリンク・レベル以下のプロトコルなど)のことで，別の表現をすると「インプリメンテーションに依存する部分」ということになります．もちろん，ISP，ESPともに期待しているハードウェアはEthernetであり，ISPではその最終的な物理層一歩手前のリンク・レベルで，デバイス・ドライバが介在します．一方，ESPでは，ハードウェア自身がIPプロトコルの一部を処理するようになっているので，ブラックボックスの処理の内容がやや異なります．

問題はこの点から発生します．ISPパッケージでは，どのようなハードウェア構成(コントローラ，DMAの振るまい方，など)にも対応できるよう論理部と物理部が分離されており，すでにたくさんの移植例があります．一方，ハードウェア自身がインテリジェントに動作する場合，ハードウェアがみずから処理するものと，処理しないものの整理，言い換えると論理部と物理部の切り分けがしばしば問題になります．この問題を説明する前に，まずISPパッケージの下位層の構造を説明します．

図2は，ISPパッケージに含まれるソケット以下の構造を，やや詳しく表現したものです．

ここでは，図1のソケット(③)にあたる部分，つまりユーザ(プログラム)インターフェースを受けもつのは，①のソケット・ライブラリです．このソケット・ライブラリの外部仕様は，先に述べたように，bsd 4.3と同等で，ISP，ESPに共通です．

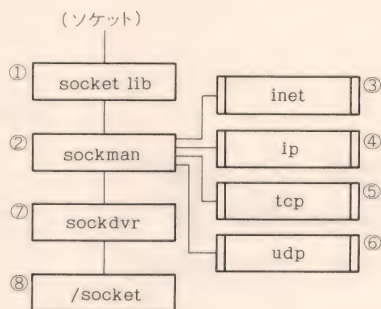
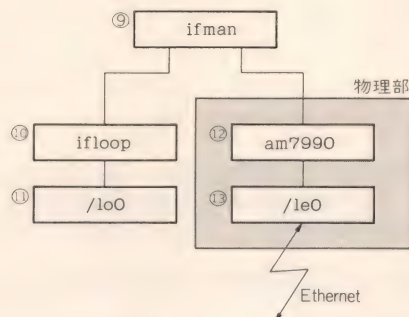
このライブラリの内部で，ソケットは/socketという名前の論理デバイスとして扱われます．したがって，②，⑦，⑧のファイル・マネージャ，デバイス・ドライバ，デバイス・デスクリプタが介在するのは，通常のOS-9デバイスと同様です．ただし，デバイス・ドライバ自体はダミーに近く，デスクリプタには，③～⑥のサブルーチン・モジュール群の名前，などが記されています．

③～⑥のinet，ip，tcp，udpは，標準のソケット・デスクリプタで指定されているサブルーチン・モジュール群です．これらのモジュール群は，システム・スタート時にメモリ上にロードされ，ソケットへのアクセス，② sockman，③ sockdvrの流れの中で呼び出され，パス・アクセスの結果として実行されます．また，これらのサブルーチン・モジュールは，ユーザ側で別の任意のものに置きかえることができますが，標準つまりTCP/IPが前提であるかぎり，その必要はありません(ただし1989年8月現在，udpはサポートされていない)．

ユーザ・プログラムからソケット・ライブラリが呼び出されると，CPUの実行スレッドは，② sockmanに移ります．ここからさらに⑦ sockdvr，③～⑥のサブルーチン・モジュールが参照されて，Internetのプロトコルとフォーマットにしたがったメッセージが組み立てられ，さらにその物理的な転送のために，CPUの実行は⑨ ifmanに渡されます．

図2(b)の⑨ ifmanは，Interface Managerです．これはOS-9の通常のファイル・マネージャで，ここでは二つのデバイス⑪ /lo0と⑬ /le0を管理しています．なお，これらのデバイスは，上記のソケット呼出しによって自動的に初期化され，この二つのデバイスにユーザ・プログラムが直接アクセスすることは認められていません．

さて，このうち⑪ /lo0とそのドライバ⑩ ifloopは，telnet自身とは直接関係のないバックグラウンド・サーバ，あるいはARPプロトコルの処理など，第3の実行スレッドでのみ使用され，これらのI/O処理は別プロセスによって再び⑬ /le0側で行われるため，⑪ /lo0と⑩ ifloop内では，物理的なI/O操作は発生

〔図2〕ISPパッケージに含まれるソケット以下の構造

(a) Internetのプロトコルとフォーマットにしたがってメッセージを組み立てるライブラリおよびモジュール群

(b) 物理的な転送を行うモジュール群

しません．

したがって，以上の ① ソケット・ライブラリから ⑪ /loO までがソケットに関する論理部分，つまり「移植」作業をすることなくすべてのシステム上で動作すべき部分です．

パッケージの最下層の物理部に，⑬ デバイス・デスクリプタ /leO と，⑫ サンプル・ドライバ am7990 が含まれています．これは MVME147 CPU ボード上の Am7990 のために書かれたもので，歴史的には以上のような論理部との分離過程を経て，ドライバ自身は与えられた Internet パケットに Ethernet ヘッダを追加し，物理的な通信とその結果の通知のみを行うようになっています．

したがって，この部分のみを他のコントローラに置きかえるか，あるいは同じ Am7990 であればポートと DMA の設計に応じて変更を加えれば，コントローラ・ハードウェアの変更(移植)はきわめて容易である，ということができます．

OS-9 の Internet パッケージ(ISP)は以上のような構成ですが，一方 ESP パッケージはやや異なります．というのは，ESP がターゲットとするのは特定の ENP-10 というインテリジェント・ハードウェアであり，簡単にいえば，ハードウェア自身がソフトウェア・プロトコルの一部をみずから処理しているからです．そのために図 2 (a)，② sockman の内部からこのハードウェア機能を呼び出すことができるように，デバイス・ドライバも，複数の実行スレッドに対して一つの特化したものが使われています．

ここまで書けばすでに明らかなように，こうしたインテリジェント・ハードウェアのもつ外部インターフェースにはさまざまなレベルが考えられます．ENP-10 の場合には，ISP での物理部の切り分けとは大きく異なる構造となり，別パッケージにならざるを得なかったわけです．さらにこの他のさまざまなインテリジェントなレベルに対応するためには，さまざまなケースを考慮する必要があり，これは必ずしも容易ではありません．もちろん，ISP パッケージに含まれるサブルーチン・モジュール(ip，tcp，udp など)をそれぞれのハードウェアに対応するものに置きかえ，ここからデバイス・ドライバを呼び出すことも考えられますが，いずれにしても，いくつかのバックグラウンド・サーバとの実行スレッドの調整は考慮しなければならず，必ずしも「移植」程度の作業で済むとはかぎりません．

現在，われわれ自身この問題は検討中であり，いずれマイクロウェアからのアナウンスがあるものと思われます．

まとめ

OS-9 のもとで，主として Ethernet を仮定した Internet 対応ソフトウェア・パッケージには，ISP，ESP の 2 種類があります．ISP は汎用の移植用パッケージであり，どのようなリンク・レベルのハードウェア(コントローラ，ボード)にも対応することができます．一方 ESP は，ENP-10 という特定のハードウェアのための専用パッケージです．

(1) これらのパッケージには，どの OS とも共通な通信とオペレーションを保証する telnet, ftp およびそれらのバックグラウンド・サーバと，Unix 4.3 bsd と同等のソケット・ライブラリが含まれている．

(2) ISP の場合，システム内部の「移植」を必要としない標準的な論理部と，ハードウェアを取り扱う物理部とに分けられ，異なるハードウェアへの移植は容易である．

(3) ただし，インテリジェントなハードウェアへの対応は，現時点で必ずしも充分ではない．

Internet をサポートする汎用ソフトウェア・パッケージがかつて商業レベルで供給されたことはなく，その意味では，ISP は画期的なものかもしれません．また，インテリジェントなハードウェアへの対応が行われれば，いわゆる Ethernet の世界にとってエポック・メーキングなものになるものと思われます．

みずの・けん

ISP/ESP のライブラリ関数

〔表A〕bsd 4.3 ソケット・ライブラリ

関　数	機　能
accept()	ソケット上の接続の受理
bind()	ソケットに名前をつける
connect()	ソケット上の接続を開始する
gethostname()	現在のホストの名前を得る
getpeername()	接続されているピア(peer)の名前を得る
getsockname()	ソケットの名前を得る
getsockopt()	ソケットのオプションを得る
listen()	ソケット上の接続をリスン(listen)する

関　数	機　能
recv()	ソケットからのメッセージを受信する
recvfrom()	ソケットからのメッセージを受信する
send()	ソケットからのメッセージを送信する
sendto()	ソケットからのメッセージを送信する
setsockopt()	ソケットにオプションを設定する
shutdown()	全二重接続の部分をシャットダウンする
socket()	通信のエンド・ポイントをクリエートする

〔表B〕OS-9 上の Internet 支援ライブラリ

Unix 上で etc ディレクトリに含まれるデータベース・ファイル(hosts, protocols, networks, services)は，OS-9 でも同様ですが，OS-9 では，これらは実行時にすべてメモリ上のデータとして保持されます．

以下の OS-9 Internet ライブラリは，このデータベースへのアクセス・ルーチン(get，set，end グループ)と，若干のユーティリティ関数から成っています．これらのルーチンは，**表A**のソケット・ライブラリを通じての通信互換性を支援するものであり，決して特殊な通信手段を提供するものではありません．また，これらの関数は，多くはソケット・ライブラリの内部で使われているにすぎません．

関　数	機　能
endhostent() endnetent() endprotoent() endservent()	ネットワーク・データベースからの Unlink
gethostbyaddr() gethostbyname() gethostent()	ネットワーク・ホストのエントリを得る
getnetbyaddr() getnetbyname() getnetent()	ネットワーク・エントリを得る
getprotobyname() getprotobynumber() getprotoent()	プロトコル・エントリを得る
getservbyname() getservbyport() getservent()	サービス・エントリを得る
htonl()	ホストとネットワークとの間で 32 ビット値をバイト順に変換する
htons()	ホストとネットワークとの間で 16 ビット値をバイト順に変換する

関　数	機　能
inet_addr() inet_lnaof() inet_makeaddr() inet_netof() inet_network() inet_ntoa()	Internet アドレスの操作
ntohl()	ネットワークとホストとの間で 32 ビット値をバイト順に変換する
ntohs()	ネットワークとホストとの間で 16 ビット値をバイト順に変換する
sethostent()	ネットワーク・ホストのエントリの設定
setnetent()	ネットワーク・エントリの設定
setprotoent()	ネットワーク・プロトコル・エントリの設定
setservent()	ネットワーク・サービス・エントリの設定
_ss_sevent()	ソケット上にイベントを設定する

リアルタイム・モジュールiREXの仕様とプログラミングの実際

木村 昭夫

OS-9/68K は,init で拡張モジュールを指定することにより,カーネル・レベルで機能を拡張できる.OS-9/68K の動作環境はそのままにして,リアルタイム性能を向上させるために実現された拡張モジュール群がiREX である.iREX の動作原理は,プロセスのスケジューリングおよびディスパッチを行うシステム・コール F$AProc および F$NProc を先取りし,OS-9/68K のプロセスより,iREX のタスクを優先させて処理するというものである.(編集部)

1 OS-9/iREXとは

OS-9/iREX はOS-9/68000 または,OS-9/68020(以下,OS-9 と総称)に iREX モジュール群を追加することで実現される拡張OS-9の名称です.iREX (industrial-Realtime-EXecutive)の名称が示すように,リアルタイム機能を充実させるモジュール群です.

OS-9/68K では,システム・コールも,システム・コンフィギュレータに開放されており,またOS9P2で知られる拡張モジュール群を init で指定することにより,カーネル自体を拡張できます.iREX はこの拡張モジュールとしてOS-9/68K システムを拡張します.つまり,OS-9/68K のこの機能を利用すれば,システム・コンフィギュレータ独自のOS-9を作成することも可能となるわけです.

〔図1〕OS-9/iREX のモジュール構成

▶リアルタイム処理の高速化

OS-9/68K のリアルタイム部分の機能強化を行ったのが,OS-9/iREX です.iREX モジュール群を OS-9/68K システムにインストールすると,図1の構造を取り OS-9/68K と iREX が一つのシステムのなかで共存します.

OS-9/68K は広範な機能をもっており,なおかつリアルタイム・モニタに比肩する速度を実現しています.しかし,システム設計に柔軟性をもたせているために,割込みの応答速度やプロセス切替え時間が,割込み要因の種類や実行待ちプロセスの数によって動的に変化します(表1).

そこで iREX では,タスク(注)数を最大256個に制限し,さらに実行待ちタスクを優先度単位にビットマップで管理することで,実行待ちタスクの数によってタスク切替え時間が変化することのない,すなわち平坦化されたスケジューラを実現しています.**表2**に示すように,OS-9/68020の約1.6倍のタスク切替え速度を実現しています.

このタスク切替えの速度向上と平坦化されたスケジューラにより,リアルタイム処理のタイミング設計が簡便になります.それは,OS-9/68K とは異なり,すべ

(注) OS-9/68K では,ユーザ・プログラムの実行状態をプロセスと呼んでいるが,iREX では,OS-9/68K と区別するためにタスクと呼ぶことにする.

〔表1〕OS-9/68K タイミング・サマリ
(マイクロウェア・ジャパン参考値)

OS-9/68000・68020 ver 2.0		
MPU	68000 12.5 MHz	68020 20 MHz
タスク・スイッチ(μs)	149+4 t	54+1.4 t

※ t はアクティブ・プロセス・キューに含まれるプロセスの数 t≧1

ての機能を実現するわけではなく，リアルタイム記述を重点的に処理するためにプログラミング・スタイルをある程度制限して，所定の速度を得ようという試みです．

▶ OS-9/68K との共存

高速性・拡張性を維持するために，iREX は OS-9/68K と共存しています．この状態を OS-9/iREX と呼びます．

OS-9/iREX では，OS-9/68K のコマンドやユーティリティのほとんどがそのまま使用できます．iREX タスクは高速性を実現し，OS-9/68K プロセスで拡張性・互換性を維持しています．

iREX タスクからは OS-9/68K のほぼすべてのシステム・コール要求を出すことができます．OS-9/68K からは，iREX タスクの生成・起動のみが許されています．iREX タスクの I/O 処理は OS-9/68K の I$ システム・コールを使用することで実現されるため，OS-9/68K のファイル・システムとの互換性が保たれます．また，パイプなどのデバイスによる同期は iREX タスクあるいは OS-9/68K プロセスの区別なく行えることになります．なお，iREX のシステム・コールは**表 3** のとおりです．

iREX と OS-9/68K を共存させるために I/F モジュールが作成されています．この I/F モジュールは iREX モジュールから OS-9/68K と認識されるもので，後述するスケジューラの振る舞いに重要な役割を果たします．

▶インストール手順

OS-9/iREX のインストールを，ここで説明します．

iREX システムをインストールする際に必要なモジュールは，① iREX モジュール，② iREX I/F モジュール，③ iREX コンフィギュレーション・モジュール (iREXinit)，の 3 モジュールです．

また，OS-9/68K の init モジュールには iREX モジ

〔表 2〕OS-9/iREX タスク切替え時間(実測値)

MC68020 (16 MHz リード時ノー・ウェイト・キャッシュ・イネーブル)		
OS のタイプ	OS-9/iREX	OS-9/68020
時間(μs)	89.3	145

〔表 3〕iREX のシステム・コール一覧

システム・コール名	機　　能
R$Addrflag	イベント・フラグ・アクセス・アドレスの獲得
R$Addrsema	セマフォ・アクセス・アドレスの獲得
R$Addrmabx	メールボックス・アクセス・アドレスの獲得
R$Addrmapl	メモリ・プール・アクセス・アドレスの獲得
R$Addrtcb	タスク・アクセス・アドレスの獲得
R$Blckget	メモリ・プールからのメモリ・ブロック獲得
R$Blckrel	メモリ・プールへメモリ・ブロック返還
R$Cyclcanc	タスク周期起床を無効
R$Excpdefi	例外処理ハンドラを定義し接続
R$Exitdefi	終了処理ルーチン定義 例外処理のタイプ指定
R$Flagcret	ID でイベント・フラグ生成
R$Flagdelt	イベント・フラグ削除
R$Flagset	ビット・パターンを以前の値との AND, OR, EXOR, または REPLACE 設定
R$Flagwait	指定ビットのすべて，またはいずれか一つが設定されるのを待つ(条件成立後のビットのクリアの有無およびタイムアウトの有無を指定可)
R$Intrdefi	ユーザの割込み処理ハンドラ定義(システムへの接続/切離し)
R$Mabxcret	ID でメールボックス生成 受信待ちの方法(FIFO・優先度) メッセージの順番(FIFO・優先度) 指定可
R$Mabxdelt	メールボックス削除
R$Meplcret	ID でメモリ・プール生成 プールのサイズ，OS の管理内か否か指定
R$Mepldelt	メモリ・プール削除
R$Mssgrecv	メッセージ受信 タイムアウト指定可
R$Mssgsend	メッセージ送信
R$Priochge	タスクの優先順位変更
R$Rdyqrote	実行待ちタスクの回転
R$Semacret	ID でセマフォ生成 初期値および待ちの方法 (FIFO または優先度順)指定可
R$Semadelt	セマフォ削除
R$Semasigl	カウント値送信
R$Semawait	カウント値待機 タイムアウト指定可
R$Stattask	タスク状態獲得
R$Taskabot	自タスク異常終了
R$Taskcret	ID でタスク生成
R$Taskdelt	タスク削除
R$Taskexdl	自タスク正常終了および削除
R$Taskexit	自タスク正常終了
R$Taskresm	強制待ち状態タスク再開
R$Taskslep	待ち状態へ遷移
R$Taskstar	タスク起動
R$Tasksusp	強制待ち状態へ遷移
R$Taskterm	他タスク強制異常終了
R$Taskwait	一定時間待ち状態へ遷移
R$Taskwkup	待ち状態タスク起床
R$Timeget	システム・クロック値読出し
R$Timeset	日付，時刻設定
R$Wkupcanc	タスク起床要求解除
R$Wkupcycl	タスク周期起床

ュールおよびI/Fモジュールを拡張モジュールとして定義しておく必要があります．ブート・モジュールに上の3モジュールを加えてOS9Genを行うことにより，OS-9/iREXが完成します．

2 iREXの内部機構

▶ I/Fモジュールのシステム・コールの先取り

図1の上段の中央にあるのがI/Fモジュールです．まずOS-9/68Kのシステム・コールの構造を簡単に説明します．システム・グローバルは，システム/ユーザ・ステートそれぞれのシステム・コールのサービス・テーブルをもちます．サービス・テーブルには256個のサービス・ルーチン・アドレスとサービス・データ・アドレスがあり，ワーク・エリアを各サービスごとに別々に定義することが可能です(図2)．

I/Fモジュールと iREXモジュールは，システム・コールの先取りを行っています．システム・コールの先取りは，システム・コール本来の処理を行う前に別ルーチンへエントリさせ，必要に応じて本来の処理ルーチンに制御を渡します．OS-9/68Kのシステム・コールのF$SSvcは，システム・コールの更新や追加を行うことができますが，古い情報は保存されないため，本来の処理を呼び出すためには不十分です．そこで，古い情報を保存してからサービス・テーブルを更新することになります．サービス・テーブルの更新は，iREXモジュールが拡張モジュールとして初期化作業を行うときに行われます．

〔図2〕システム・コール(システム用)の構造

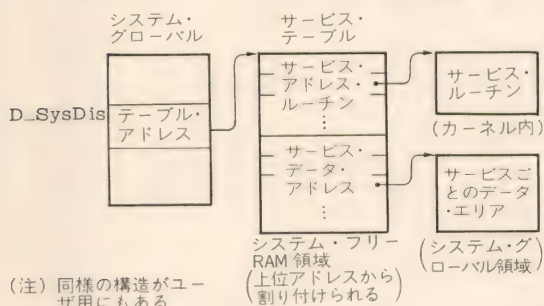
I/Fモジュールが先取りするシステム・コールは，F$AProc(スケジューラ)とF$NProc(ディスパッチャ)の二つのシステム・ステートのシステム・コールです．これらのシステム・コールは，iREXモジュールも先取りしており，I/Fモジュールは必ずiREXモジュールから呼び出されます．それぞれのシステム・コールは，つぎのとおりです．

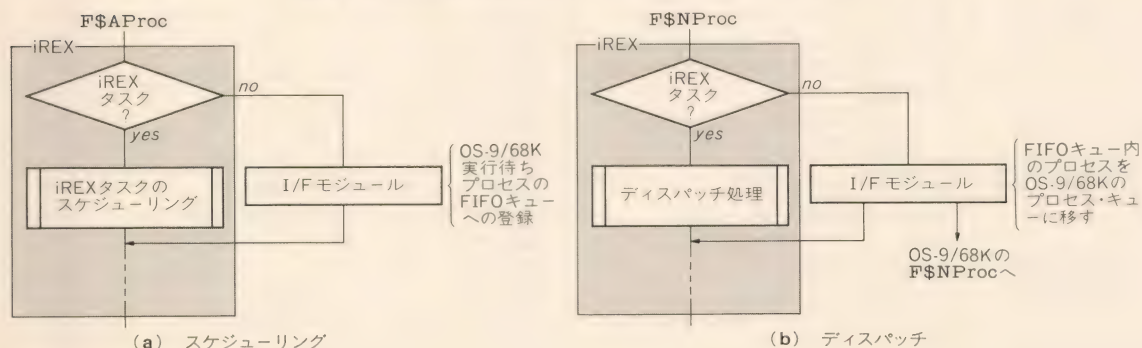
(1) **F$AProc**

OS-9/68Kプロセスが実行待ち状態へ遷移するときに呼び出されます．I/Fモジュールでは，実行待ちプロセスのFIFOキューを作成しており，そのキューへ実行待ちプロセスを挿入します．本来のOS-9/68KのF$AProcのサービス・ルーチン(スケジューラ)を呼び出すことはありません(図3(a))．

(2) **F$NProc**

実行すべきiREXタスクが存在しないときにiREXモジュールより呼び出されます．実行待ちのOS-9/68Kプロセスが存在しない場合は，キャリ・フラグをセットして戻ります．実行待ちプロセスが存在すれば，OS-9/68KのF$AProcに相当する処理を行い，実行待ちのOS-9/68Kプロセスすべてのスケジューリングが終了した時点でOS-9/68KのF$NProcへエントリします(図3(b))．

OS-9/iREXでは，システム・ステートにおけるタイムアウト・ディスパッチは認めていません．したがって，1回の処理が長引くとシステムの最悪応答時間が長くなってしまいます．そこで，このF$AProcに相当する処理(スケジューリング)は，4フェーズに分割されていて，1回の呼出しでは1フェーズだけ処理します．4フェーズとは，① FIFOキューからの実行待

〔図3〕OS-9/iREXにおけるスケジューリングとディスパッチ

(a) スケジューリング　　(b) ディスパッチ

ちプロセスの取出し，② エージング処理，③ アクティブ・キュー挿入場所のサーチ，④ 挿入，です.

さらに，1フェーズの中で繰返し処理が発生する場合には，1回の呼出しでは繰返し処理の1回ぶんだけ実行し，繰返し状態を保持したまま，呼出し側(iREXモジュール)に戻ります.

▶スケジューラ

iREXでは，タスクは256個まで作成することができます．また優先度も1から255まで指定可能で255が一番高い優先度になります．OS-9/68Kのプロセスは，iREXタスクが実行されていないときだけ動作することができます．つまりiREXタスクの最低の優先度のタスクであってもOS-9/68Kの最高優先度のプロセスより優先的に実行されます．iREXタスク・グループとOS-9/68Kプロセス・グループがあり，iREXタスク・グループが最優先に実行されるイメージです．これは，iREXタスクの応答速度やタスク切替えの速度を維持するための設定です.

OS-9/68Kのプロセスは，iREXタスクが実行されていない間だけ，OS-9/68Kのスケジューリング手法にもとづいて実行されます．iREXがOS-9/68Kスケジューリング手法の影響を極力受けないように，I/Fモジュールのディスパッチ・ルーチンは処理を数フェーズに分割してあり，1フェーズごとにiREXへ戻ります．こうして少しずつ処理を行い，それが終了した時点でまだiREXタスクがアクティブでないときに，I/FモジュールがOS-9/68Kのディスパッチ・ルーチンへ制御を渡すことで，OS-9/68Kのプロセスが動作できます(図4).

OS-9/iREXではプリエンプションを採用しているため，OS-9プロセスが実行中でもiREXタスクがアクティブになった時点で，iREXタスクへ実行権が割り当てられ，OS-9/68Kプロセスは実行待ち状態になります．iREXのスケジューリング手法は，優先度ベースで行っています．同一優先度のタスクの実行順位は到着順とし，タイム・スライスによるタスク切替えは発生しません．ただし，システム・コールにより実行待ち状態のタスク・キューをローテートすることができます.

〔図4〕iREX側のフロー

図5に示すように，実行待ち状態のタスクは，各優先度ごとに双方向のリスト構造をもっています．タスクの開始や事象の成立などで実行要求が出された場合，そのタスクの優先度からキュー・テーブルを決定してキューへの挿入を行った後，ビットマップをセットします．キューは各優先度ごとに独立して用意されているため，優先度にしたがって自動的にアドレスが決定されます．このキューは円環構造をもっているために実行要求順のキューへの挿入が一定時間で行えます.

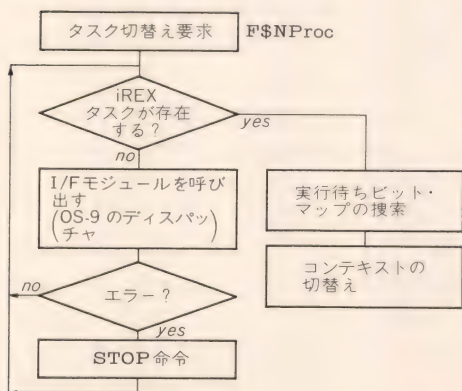
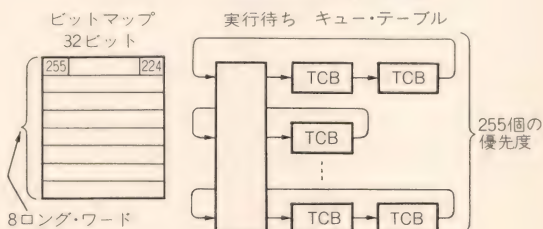
つぎに実行すべきタスクの決定は，ビットマップを優先度の高い順に探索していきます．実行待ち状態のタスクが存在する優先度はビットマップがセットされているので，ビットマップがセットされている部分まで行います．ビットマップを探索し，すべてが0であれば，実行待ちタスクがないことになります．優先順位が決定した後は，1回のアクセスでキューの先頭のタスクを取り出すことができます.

▶タスク間の同期・通信

タスク間の同期・通信処理を行うために，iREXではイベント・フラグ，セマフォ，メールボックスの機能が用意されています.

OS-9/68Kでは，F$Eventと呼ばれるイベント・セマフォが提供されていますが，iREXでは，サービスを行う最大個数を制限することで，高速性を追及しています．サービスの個数は，iREXinitモジュールにそれぞれのサービスごとに定義されますが，それぞれ256個を越えることはできません.

(1) イベント・フラグ

イベント・フラグは，事象の発生の有無を1ビットのフラグで管理することによりタスク間の同期を行い

〔図5〕OS-9/iREXの実行待ち状態

ます．２進セマフォと呼ばれるものに相当します．iREX では，このフラグを 32 個まとめて１イベント・フラグとして管理します．タスクは，32 個の事象を AND または OR で組み合わせて，待つことができます．

OS-9/68K では，複数のプロセスが同一イベントで事象待ちできました．iREX では一つのイベント・フラグで待つことができるのは１タスクだけに制限されています．またイベント待ちでは，タイムアウトの指定ができます．事象が成立したときには，イベント・フラグの当該ビットをセットして待ちタスクに通知することになります．

(2) セマフォ

OS-9/68K の F$Event のサブセットになります．iREX のセマフォのもつ機能は，OS-9/68K の Ev$Wait，Ev$SetR(Ev$Incr)の二つに相当します．セマフォの場合と同様，タイムアウト機能が追加されています．これにより，一定時間待ってもフラグがセットされない場合に，別の処理をできることになります．

(3) メールボックス

メールボックスは，iREX で追加された機能でデータの受け渡しに使われます．

OS-9/68K では，データ転送の手法としては，(名前付き)パイプ，データ・モジュールが用意されています．パイプはデバイスですし，データ・モジュールは同期手法を別に用意しなければなりません．

iREX のメールボックスは，メッセージそのものを渡すのではなくメッセージのアドレスを渡します．メッセージは，受信タスクによって参照されます．この方法では，システムがメッセージそのものを管理しないため，送信タスクはメッセージ・エリアを保持する必要がありますが，**図 6** に示すようにシステムによる転送の必要がなくなります．

メールボックスは，メールボックスの ID によってメッセージの送受信を行います．メッセージを送信するとただちにキューイングされます．メールボックスにまだメッセージが届いていない場合には受信要求タスクは休止されます．

〔図６〕メッセージの送信手法

メッセージ受信要求の場合には，タイムアウトの指定ができます、一つのメッセージは，0 から 31 の優先度をもつことができ，31 が優先されます．または，要求順の指定もできます．

このメールボックスの利点を，他の手法と比較してみましょう．

パイプの場合，ユーザ・タスクは同期のことは意識せずに，データ転送ができますが，データ転送の際，データ待ちでタスクが停止したり，受取りタスクを待ったりします．データの同期待ちをしている間に，他の処理ができれば，効率はよくなります．メールボックスは，送信タスクが受信タスクを待つことはありませんし，受信タスクもタイムアウトを使用することで別の処理に切り替えることもできます．

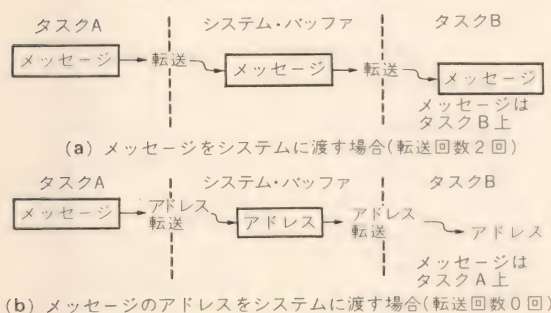
データ・モジュールは，送受信側とも同期を意識しなければなりません．データ転送の領域はデータ・モジュール内に取られるので，ユーザは領域確保の必要がありませんが，その半面データ・モジュール内の領域が使用中であれば，領域が空くまで待たなければなりません．

メールボックスは，送信タスクの責任において送信領域を確保します．メモリが許すかぎり受信タスクを意識せずにメッセージを送りつづけることができます．

▶メモリ管理

メモリ管理のためのシステム・コールも追加されています．メモリ管理のために，メモリ・プールの概念を採用しました．メモリ・プールは，タスクが将来使用するであろう領域をまえもって確保しておき，必要に応じ確保した領域に対してのメモリ要求・返還を行うものです．

OS-9/68K では，使用されていないメモリ・エリアはフリー RAM リストとよばれるリスト構造で管理されています．メモリ要求が行われると，このフリー RAM リストに要求されたサイズの連続領域が存在するかを探索します．もし存在すれば，この領域をフリー RAM リストより抜き出し，ユーザ・プロセスに与えます．必要になった時点で，初めてメモリ要求を行うために，基本的に無駄なメモリ要求は発生しません．ただし，同じ理由で，メモリ獲得時間はフリー RAM リストの状態によって変化してしまいます．

メモリ・プールからのメモリ・ブロックの獲得は基本的に固定長のメモリ要求となるために，メモリ獲得時間は平坦化されます．メモリ・プールの生成時点では，iREX は OS-9/68K に対してメモリ要求を行います．これは，メモリ資源の有効利用のための措置です．

メモリ・プールの作成は，タスクが起動直後に行うものと仮定しています．つまりタスク起動直後は，まだタイミング的に難しい処理や，同期を必要とするような状況にはまだ達していないため，実行時間が平坦化されていない処理は，極力起動直後に行います．タスクの同期部分の処理時間の算出を容易にします．

メモリ・プールの生成時には，メモリ・プールの構成単位であるブロックを定義します．メモリ・プールよりメモリを要求する際には，何ブロック要求するかを指定します．つまり，ブロック長などで，ある程度の柔軟性をもたせることはできますが，基本的には固定長のサービスになります．固定長は柔軟な使用法には向きませんが，メモリの獲得時間は非常に短縮されます．メモリ・プールで連続したメモリを獲得しているため，タスクの実行途中でのメモリのフラグ・メンテーション（分断）によるメモリ獲得失敗がなくなりま

〔リスト１〕親タスク

```
#include    <irex.h>
#include    <errno.h>

#define     ERROR       (-1)
#define     DELAY_COUNT 0x1000000

#define     CHILD_ID    15
#define     MAIL_ID     16
#define     FLAG_ID     17
#define     MAIL_PRTY   30
#define     ACK_PTN     0x00000002          /* bit1 のセット */

#define     CHILD_NAME  "if_child"

void delay_loop(count)
register int count;
{
    while(count-->0);
}

void initial_resorce()
{
    u_short option = 0;   /* 送•受信ともに優先度キューイング */
    short   msgmax = 32;  /* 最大メッセージ数 128 */

    /*      メールボックスを作成する        */
    if (_mabx_cret(option, MAIL_ID, msgmax) == ERROR) {
        exit(_errmsg(errno,
                    "Can't Create Mailbox[%d]. ¥n", MAIL_ID));
    }
    /*      イベント•フラグを作成する        */
    if (_flag_cret(FLAG_ID) == ERROR) {
        exit(_errmsg(errno,
                "Can't Create event_flag[%d]. ¥n", FLAG_ID));
    }
}

void terminate_resorce()
{
    /*      メールボックスを削除する        */
    if (_mabx_delt(MAIL_ID) == ERROR) {
        exit(_errmsg(errno,
                    "Can't Delete Mailbox[%d]. ¥n", MAIL_ID));
    }
    /*      イベント•フラグを削除する        */
    if (_flag_delt(FLAG_ID) == ERROR) {
        exit(_errmsg(errno,
                "Can't Delete event_flag[%d]. ¥n", FLAG_ID));
    }
}

start_child(child_id)
register u_short child_id;
{
    u_long  addstack = 0;
    u_short priority;
    short pathcnt;
    char *modname = CHILD_NAME; /* タスク名 */

    priority = 0;   /* タスク優先度 親タスクと同一 */
    pathcnt = 3;    /*  I/O パス数 */

    if (_task_cret(child_id, modname, addstack,
                            priority, pathcnt) == ERROR) {
        exit(_errmsg(errno,
                    "Can't Create task[%d]. ¥n", child_id));
    }
    printf(" 子供タスクを起動します。 ¥n");
    if (_task_star(child_id, "dummy") == ERROR) {
        exit(_errmsg(errno,
                    "Can't Start task[%d]. ¥n", child_id));
    }
}

u_char *
get_msg()
{
    u_char *messagep;

    if ((int )(messagep = (u_char *)_mssg_recv(0, MAIL_ID, 0))
                                            == ERROR) {
        exit(_errmsg(errno,
            "Can't receive message mailbox[%d]. ¥n", MAIL_ID));
    }
    return (messagep);
}

set_ack()
{
    u_long  p_patn;

    if (_flag_set(T_OR, &p_patn, FLAG_ID, ACK_PTN) == ERROR) {
        exit(_errmsg(errno, "Can't set flag[%d]. ¥n", FLAG_ID));
    }

}

main()
{
    register u_char *msgptr;
    initial_resorce();
    start_child(CHILD_ID);
    delay_loop(DELAY_COUNT); /* 時間潰し */
    msgptr = get_msg();
    printf("%08x is %s ¥n", msgptr, msgptr);
    set_ack();              /* 子供への受信通知 */
    sleep(1);
    terminate_resorce();
    printf(" 親タスクは終了しました。 ¥n");
}
```

す．これもまた重要な機能といえます．

3 iREX の実際

iREX のメールボックスとフラグを使用した簡単なプログラムを**リスト 1** および**リスト 2** に示します．また，makefile を**リスト 3** へ示します．

ここでは便宜的に，**リスト 1** を親タスク，**リスト 2** を子タスクとしてあります(iREX タスク間には親子関係はない)．

(1) 親タスク(**リスト 1**)

- イベント・フラグ，メールボックスの作成
- 子タスクの生成，起動
- ディレイ・ループ

〔リスト 2〕子タスク

```
#include    <irex.h>
#include    <errno.h>

#define     ERROR       (-1)
#define     DELAY_COUNT 0x1000000

#define     CHILD_ID    15
#define     MAIL_ID     16
#define     FLAG_ID     17
#define     MAIL_PRTY   30
#define     ACK_PTN     0x00000002  /* bit1 のセット */


u_char *message = " これは子供タスクのメッセージです。";

void delay_loop(count)
register int count;
{
    while(count-->0);
}

void send_parent(msgp)
register u_char *msgp;
{
    if (_mssg_send(MAIL_ID, msgp, MAIL_PRTY) == ERROR) {
        exit(_errmsg(errno,
                "Can't send message mailbox[%d]. ¥n", MAIL_ID));
    }
}

wait_ack()
{
    register u_short option = (T_ANDW | T_RESET);
                                    /* AND 指定・リセット有り */

    u_long  p_patn;

    if (_flag_wait(option, &p_patn, FLAG_ID, ACK_PTN, 0)
                                                    == ERROR) {
        exit(_errmsg(errno, "Can't wait flag[%d]. ¥n", FLAG_ID));
    }
}

main()
{
    printf(" 子供 : タスクが起動されました。¥n");
                                        /* 開始メッセージ */
    printf(" 子供 : メッセージアドレスは、%08x です ¥n",
                                                message);
    delay_loop(DELAY_COUNT); /* 時間潰し */
    send_parent(message);    /* 親タスクへのメイルの送信 */
    wait_ack();              /* 親タスクの ACK 待ち */
    printf(" 子供タスクは終了しました。 ¥n");
}
```

〔リスト 3〕makefile

```
#
# Makefile to iREX C-libraries sample programs
#

#
# Object names
#
OFILES      = if_parent if_child

#
# ROF names
#
RFILES      = if_master.r if_parent.r if_child.r

#
# Directories definition
#
ODIR        = OBJS
WDIR        = /dd
DEFSDIR     = /dd/defs
RDIR        = RELS
LIBDIR      = /dd/lib
STBDIR      = STB

#
# commands name definition
#
CC          = icc
OC          = o68
DEL         = del
LC          = il68
ECHO        = echo
ATTR        = attr

CSTART      = $(LIBDIR)/icstart.r

#
# commands options definition
#
CFLAGS      = -gqxi -t=$(WDIR) -r=$(RDIR)
LFLAGS      = -l=$(LIBDIR)/irexlib.l¥
              -l=$(LIBDIR)/sys.l¥
              -l=$(LIBDIR)/irexsys.l¥
              -l=$(LIBDIR)/clibn.l¥
              -g

RFLAGS      = -q
ATTR_FLAG   = -apee

make.date: $(OFILES)
    touch $@

$(OFILES) : $(LIBDIR)/cio.l $(LIBDIR)/clib.l $(LIBDIR)/sys.l
  $(LC) $(LFLAGS) $(CSTART) $(RDIR)/$@.r -O=$(ODIR)/$@
  (chd $(ODIR);¥
  $(ATTR) $@ $(STBDIR)/$@.dbg $(STBDIR)/$@.stb $(ATTR_FLAG);¥
 )

$(RFILES) :
    $(CC) $(CFLAGS) $*.c                    #normal mode
```

●メッセージ受信
●メッセージ表示
●メッセージ受信のフラグをセット
●実行時間の放棄(これにより子タスクが実行される),メールボックス, イベント・フラグの削除
●終了メッセージ
(2) 子タスク(**リスト 2**)
●起動メッセージの表示
●メッセージ・アドレスの表示(実行コード中)
●ディレイ・ループ
●メール送信
●受信フラグ待ち
●終了メッセージ表示

実行結果を**図 7** に示します.

4 iREXの開発環境

iREX は, OS-9/68K を拡張しているために OS-9/68K のほとんどすべてのコマンドが使用可能です. ただし, これらのコマンドは OS-9/68K プロセスとして動作することに注意してください.

iREX システム・コールを使用するオブジェクトを作成するために OS-9/68K のコマンドに加え, 以下のものを使用します.

icc	iREX 用 cc ドライバ
ir68/020	iREX 用アセンブラ
il68	iREX 用リンカ
ilexsys. l	iREX 用システム・ライブラリ
irexlib. l	iREX 用 C ライブラリ
icstart. r	iREX 用 cstart

ただし, iREX システム・コールは後述するようにトラップ番号とシステム・ライブラリが異なるだけなので, OS-9/68K の標準コマンドだけを使用しても作成することができます.

icc は iREX 用の cc ドライバです. 標準の OS-9/68K の C ライブラリの他に iREX 用のライブラリ (irexsys. l, irexlib. l)をリンクします. また, iREX 用のアセンブラ, リンカを呼び出します.

ir68(020)は, OS-9/68K のアセンブラに対して, 以下の機能が追加されています.

● irex 命令…これは, r68 の OS9 命令と同じ形式で, iREX のシステム・コールが記述できます.
● align2 命令…align 命令がワード長(2バイト)で整合するのに対しロング・ワード長(4バイト)で整合します. 整合処理のためのオプション i が追加されています. 機能としては, 奇数ワード長命令の部分に nop 命令を挿入することで, 命令境界を, ロング・ワード長(4バイト)に整合します. 整合のモードには, ① モード 1 (すべての奇数ワード長命令に対して, nop 整合を行う), ② モード 2 (ラベルをもつ奇数ワード長命令に対して, nop 整合を行う), ③ モード 3 (大域ラベルをもつ奇数ワード長命令に対して, nop 整合を行う)の 3 種類があり, ユーザが指定することができます.

〔図 7〕実行結果

```
子供：タスクが起動されました。
子供：メッセージアドレスは、000cb81c です
子供タスクは終了しました。
子供タスクを起動します。
000cb81c is これは子供タスクのメッセージです。
親タスクは終了しました。
```

il68 は, MC68020 以降の CPU 向けにモジュール・サイズやモジュール・エントリをロング・ワード(4バイト)単位に整合させる他は, OS-9/68K の l68 とまったく同一です.

また, iREX 用のデバッグ・コマンドとして isysdbg があります. isysdbg は OS-9/68K のシステム・ステート・デバッガ sysdbg を iREX 用に変更したものであり, ブレーク・ポイントの設定, シンボル・テーブルの使用など, sysdbg とまったく同一に使用できます.

OS-9/68K でのユーザ・モード・デバッガ debug は OS-9/68K の親子関係を使用しているため, iREX では, 使用できません.

きむら・あきお ㈱マイクロボード

ニュースの背景

1億ドルの出資で，NeXTの販売権を取得したキヤノンの思惑

オブジェクト指向，マルチメディア対応など先進機能をふんだんに盛り込んだ次世代ワークステーション，The NeXT Computer System(以下，たんにNeXTと略す)が9月から日本でも販売されることが本決まりになった．販売権を獲得したのはNeXT用の光磁気ディスク，レーザ・ビーム・プリンタを製造するキヤノンである．キヤノン本社内にNeXT専門の販売部隊を設け(系列のキヤノン販売ではない！)，当初は英語版，そして1990年秋をめどに日本語版NeXTも投入する計画だ．

NeXTといえば，米国コンピュータ業界が生んだスーパースター，スティーブン・ジョブズ氏が産みの親である．キヤノンは同氏の会社，ネクスト社に1億ドル(約140億円)も出資をしたうえで販売権を獲得した．これだけの投資は「前例がない」(キヤノン)だけに，キヤノンとしては是が非でもNeXTを日本市場に定着させたいところだ．だが，日本のワークステーション市場には日本サン・マイクロシステムズやHP，ソニー，日本電気といった強豪がひしめくうえ，アップルコンピュータのMacintosh IIcxも競争相手になる激戦区である．NeXTがどこまでシェアを伸ばすか，競合メーカの販売担当者たちは興味津々で見守っている．

CAD/CAMなど実務用途ではNeXTは不利

では，実際のところNeXTはどれだけ売行きを伸ばせるのだろうか．現在，ワークステーションの主力ユーザは大学などの研究機関，ソフトハウス，企業の研究開発部門などだ．用途はソフトウエア開発，シミュレーションや実験結果の整理，それにCAD/CAMといったところが大半を占める．つまりワークステーションを遊びではなく，実用に徹して使うところが多いわけだ．

こうしたユーザが導入するマシンを決める際，重要な要件は二つある．①ハードウェアのコストパフォーマンスが高いこと，②流通アプリケーションが数多く動くこと，である．これらの観点からNeXTを見てみよう．

まずハードウェアのコスト・パフォーマンスではNeXTは相当苦しい立場に追い込まれる．サン・マイクロのSPARC Station/1やDEC社のDEC Station 2100に代表されるように1MIPS当たりの価格が10万円程度のものが増えつつある．これに対しNeXTはMPUが68030(25MHz)，約7MIPSで200万円(英語版)もする．しかもNeXTのディスプレイは4階調とはいえモノクロ．CAD/CAMをはじめとする実用的なアプリケーションを稼働させるには物足りない．標準装備の光磁気ディスクや音声処理機能は魅力だが，実務に使うにはフロッピ・ディスクがないなど？マークが付く．

アプリケーション・ソフトウェアについても新参者であるNeXTが不利で，サン・マイクロの3000本以上の蓄積にはとうてい追いつけない．もちろん主要なソフトはそう遠くないうちに移植されるだろうが，主要なものだけでは勝負にならないことはこれまでの歴史が証明している．

売りもののユーザ・インターフェースと，ユーザ・インターフェース構築ツールNextStepにしてもこれによってマシンが売れるほどのものではない．もしそうならSUNワークステーション用のSmalltalk-80がもっと売れていてもいいはずだからだ．

こう見てくると実務にワークステーションを使うユーザーがNeXTを買うと考えることは，かなり無理があることがわかる．大学の情報関係学部や一部のソフトハウスなどが趣味やNeXTそのものの研究用として購入することは十分考えられるから，2000から3000台程度は売れても，日本語処理ができない，「アカデミック・ディスカウントは日本ではしない」(ジョブズ社長)ため，日本サンやソニーのシェアを脅かすほどの存在にはなりそうもない．

キヤノンはNeXTをどう売るか

「富士通がCD-ROM搭載のTOWNSで，日本電気に挑戦したが，うまくいっていない．これと同じことがネクストとサン・マイクロ，ソニーについていえるのではないか」ある業界有力筋はこう指摘する．マシンのレベルや狙う市場は異なるものの，あながち的はずれとはいえない．

もっともTOWNSとNeXTには米国市場に基盤をもつかどうかという大きな違いがある．仮に米ネクストが米国での販売を伸ばし，NeXTのアプリケーション・ソフトや周辺ハードの蓄積を進めれば，それが日本での展開にも好影響を及ぼすからだ．実際，ネクストは米コンピュータ流通大手，ビジネスランド社と提携，その一方で米IBMにNextStepをライセンス供給するなど米国市場で好スタートを切っている．そのためベータ・テスト段階にすぎないこれまでのNeXTでさえ，すでに3000台以上を出荷，「NeXTの完成版が出荷される8月以降，販売は大幅に伸びる」とジョブズ社長は自信を強めている．

ただ，米国市場では競合メーカ，中でもアップルが“ネクストつぶし”に出るという噂がある．「ジョブズ氏がアップルをやめる際に競合製品は作らない，と約束したにもかかわらず，NeXTは教育，ビジネス両市場でアップル製品と真っ向から競合するからだ．事実，アップルとネクストの関係の悪さはアドビ社をめぐる一連の動きにともなって表面化しつつある．ネクストとアドビがDisplayPostscriptで関係を強化する一方で，これまで蜜月時代を過ごしてきたアップルとアドビの仲が急速に冷えているのだ．

そうなれば米国市場でのNeXTも期待できないかもしれない．だからといってキヤノンが困難な状況に陥ることはないだろう．たとえばNAVI，AIノートといった操作性重視の電子文房具，それにHPと提携してオブジェクト指向操作環境NewWaveを日本語化するなど，いままさに花開こうとしているオブジェクト指向，マルチメディア時代を先取りしているのがキヤノンであり，NeXTの販売はその一環にすぎないからだ．さらにレーザ・プリンタでの独走的なシェア，写真なみの品質を実現したカラー・コピアDIOなど，メーカとしてのキヤノンのユニークさは際だっている．NeXTをどれだけ販売できるかは未知数だが，キヤノンの行動から目が離せないことは確かだ．　**(魚眼子)**

■私のユーティリティ・プログラム

Turbo C による ポップアップ・ウィンドウ作成用ライブラリ

後藤 龍司

アプリケーションを作成する際，ユーザ・インターフェースをよりよいものにするために，ポップアップ・ウィンドウを利用したいことがある．最近では，専用のライブラリ関数がコンパイラに付属していたり，サード・パーティから市販されたりしており，そのようなユーティリティを作成することが容易になっている．ここでは，そのようなユーティリティ作成ツールとして Turbo C の画面制御関数を利用したポップアップ・ウィンドウ作成用ライブラリを紹介する．このライブラリにより，メニュー選択用ウィンドウおよび文字列入力用ウィンドウを作成できる．　　　(編集部)

はじめに

近ごろのパソコン用ソフトは画面表示に凝っているものが多くなり，わかればよい式のソフトを納めるとイヤミの一つも言われるようになってしまいました．むしろ画面さえある程度よくできていれば中身のほうはそれなりに適当でもよい場合さえあるようです．

ROM 化するのであれば表示のことなど考えなくてもよかったのですが，パソコン用のソフトとなると画面表示とアルゴリズムが結びついてくることも多いので，設計の段階で表示についても考えておかなくてはなりませんが，これがけっこう面倒で手間取ります．とくにあまり大きくないソフトを何本か作るような場合には，画面設計が大半を占めてしまうことさえあります．

というわけで，汎用的に使える簡単なポップアップ・ウィンドウを作ってみました．

ウィンドウといっても画面設計の手間を省くために作ったようなものですので，複雑なことはできません．そのかわり小型，軽量，使用方法も簡単ですので，いろいろなソフトに組み込むことができ，けっこう重宝しています．

プログラマからみたポップアップ・ウィンドウの長所というと，何といっても画面設計が楽になることです．ウィンドウを使わない場合には文字入力にしてもどこで入力させるかを考えなくてはなりませんし，エラー・メッセージの表示場所なども決めなくてはなりません．ポップアップ・ウィンドウを使えば文字入力もメッセージ表示なども必要なときに好きな位置へ出すことができます．

これから紹介するポップアップ・ウィンドウ関数は，① メニューを選択させるタイプのものと，② ユーザに文字列を入力させるタイプのものの 2 種類です．前者は wi_select という関数で実現し，後者は wi_string という関数で実現します．どちらもテキスト画面だけを使用しグラフィックは使いません．

なお，ポップアップ・ウィンドウを実現するために使った関数群は Turbo C と PC-9801 専用のものが多く，他のマシンへのあるいは他のコンパイラでの移植性はほとんどありません．

ポップアップ・ウィンドウ

▶ wi_select 関数

wi_select 関数は，図 1 のようなタイプのメニュー選択用のポップアップ・ウィンドウで，↑キー，↓キーで選択バーを移動させリターン・キーを入力するか，メニュー左端の数値を直接入力して選択します．

後述するように，ウィンドウを作成するための各種パラメータを構造体で定義するようにしていますが，そのパラメータの一つである input フラグを ON にした場合には，メニューの中のエントリを選択バーで選択することも，キーボードから文字列を入力し，後にそれをヒストリ・データとしてカーソル移動キーを使用して選択することもできます．

上下の反転した部分にはガイダンスなどが表示できるようになっています．ガイダンスなどがない場合には図 2 のようなウィンドウになります．また，表示文

字はセンタリング・フラグを立てることにより枠の中心にもってくることができます．

カラーは，枠，メニュー文字列，選択バーを別々に指定します．

枠の形はMIFESタイプと**図1**のような形の2種類からどちらかを指定します．

▶ wi_string 関数

wi_stringは**図3**のようなポップアップ・ウィンドウをオープンし，ユーザに文字列をウィンドウ内で入力させるための関数で，入力させる文字数や文字コードの範囲を指定することができます．枠やカラー指定などはwi_select関数と同じです．

wi_selectとwi_stringはほとんど同じですので1本にまとめることも可能ですが，アプリケーションによりチョコチョコ改造して使う場合やりにくいので，2本に分けたままにしてあります．

このウィンドウ作成ルーチンでは多数のTurbo C専用の関数を利用していますので，他のコンパイラに移植するのは多分容易ではないでしょう．いちいち解析して移植するくらいならTurbo Cを買ってしまったほうが早いかもしれません．

Turbo Cの画面制御用関数

今回のウィンドウ・ライブラリで使っているTurbo C固有の画面制御用関数について簡単に説明しておきます．これらの画面制御関係の関数はcprintfやcputsなどのVRAM直接アクセス関数に対して有効になります．

● window(left, top, right, bottom)

この関数は，画面にウィンドウを設定するために使います．この関数を実行後に文字列を表示させると，指定したウィンドウの範囲内に表示され，スクロールもウィンドウ内だけで行われます．スクロールで気をつけなくてはならないことは，ウィンドウ右下の座標に書き込むと1行スクロール・アップしてしまい，最下行が空いてしまいます．たとえばウィンドウを画面いっぱいに切って，80桁25行目に書き込むと画面全体がスクロールすることになります．

これを防ぐにはそこを使わないか，80桁25行目だけVRAMに直接pokeするなどの方法をとる必要があるようです(うまいやり方があったら教えてください)．

● clrscr()

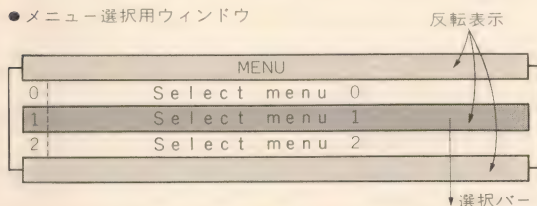
clrscr()は，window()で指定した範囲の表示を消します．

● gotoxy(colum, line)

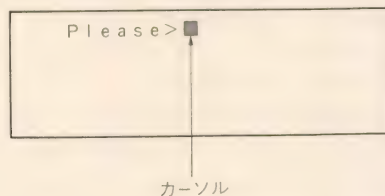
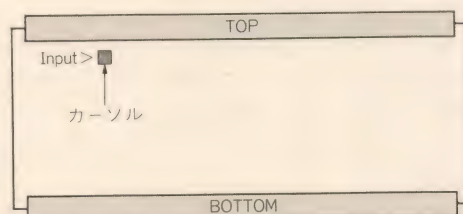
gotoxy(colum, line)は，カーソルを指定位置に移動します．window()もこの関数も，桁や行が1から始まります(0ではない)．ライブラリを見るとわざわざ内部で−1しているようですが，筆者の好みからすれば0から初めてもらいたいところです(マクロを使えばすむことではあるが)．

それからこの gotoxy()関数で指定する座標は絶対

〔図1〕wi_select関数で作成できるウィンドウの種類

●メニュー選択用ウィンドウ

●inputフラグをONにした場合(上枠だけガイダンスあり)

●下枠だけガイダンスあり

〔図2〕wi_select関数による文字列入力用ウィンドウ
(ガイダンスがなく枠だけのタイプ)

〔図3〕wi_string関数による文字列入力用ウィンドウ
(上下にガイダンスがついたもの)

座標ではなく，window( )で指定した左上の座標が 1 桁目の 1 行目になります．たとえば window(10, 10, …)と指定して gotoxy(1, 1)とやるとカーソルは画面上の 10 桁 10 行目へ行きます．

● textreverse(REVERSE/NOREVERSE)

● textcolor(T_???)

● textcursor(DISP_CURSOR/NODISP_CURSOR)

textreverse は，この関数実行以後の表示を反転/非反転させます．textcolor は同じく色を変化させます．textcursor はカーソルの表示/非表示の指定です．

● textmode(VL8025 など)

textmode(VL8025)は画面を 80 桁 25 行で使用し，PC-9801 のもつ簡易グラフィックを無効にします．textmode(BG8025)では逆に簡易グラフィック機能を有効にします．これを有効にしないとグラフィック・コードを表示させることができません(ウィンドウの枠を書くのに使用している)．そのかわり漢字が表示できなくなります．漢字を表示させるには textmode (VL8025)とする必要があります．

● gettext( )/puttext( )

gettext( )は指定範囲の画面データを取り込みます．puttext( )は gettext( )で取り込んだ画面データを指定位置に表示します．ここでは movetext( )関数は使っていませんが，ある画面範囲の表示をよそに移すような場合に使えば，gettext( )と puttext( )を使うより簡単にできるようになっています．

なお，gettext( )では画面データを取り込みますが，取り込まれた表示が消えるわけではないので，必要なら取り込んだ後で消さなくてはなりません．取り込むためのメモリはここでは malloc( )で確保していますが，取り込む範囲×4 倍以上のメモリが必要です．

また，gettext( )しないで puttext( )したり，取り込んだ以上のデータを puttext( )したりすると画面がグチャグチャになりビックリします(キレイだという人もいますが…)．

● gettextinfo( )

gettextinfo( )は画面属性の取得を行います．textcursor( )や textreverse( )で指定した属性や，VRAM のバンク，色，ウィンドウ範囲などの情報を読み出します．ここでは save_textinfo( )，load_textinfo( )という関数を作ってその中で使用しています．wi_select ( )や wi_string( )関数から，それを呼んだ側に戻る際に画面情報を戻してやるために使っています．

プログラムについて

画面制御以外の特殊な関数についてはそのつど説明していくこととして，ウィンドウ・ルーチン自身の動作を追ってみます．

前述したように，選択ウィンドウ関数 wi_select( )と文字列入力用関数 wi_string( )とは似たようなものですので，ここでは文字列入力用関数 wi_string( )について説明します．

wi_string( )を利用するためには，構造体 _wi(**リスト 1**)の各メンバに値を設定してやります．

リスト 2 を見ていただければおわかりになると思いますが，メンバがけっこう多いので毎回設定するのは面倒です．それに使い方によっては毎回設定する必要のないメンバもありますので，普通はメンバに値を設定するための関数を一段かましてやります．

たとえば何かメッセージを表示して yes か no を選択するための関数として**リスト 3** (p.232)のようなものを作っておきます．この関数内ですべてのメンバに値を設定してやり，関数自身には引数としてウィンドウ内に表示させたいメッセージだけを渡します．もしウィンドウの表示場所をそのつど変えたいのであれば，引数として座標も渡せばよいでしょう．

```
char disp_yn_mess(char *message,
                  short x, short y); など
```

このような形の「かまし関数」を何個か作っておけばメンバへの値設定の手間が省けますし，ソースも短めになります．

▶ wi_string( )関数の処理の流れ(図 4 参照)

(1) パラメータのチェック

ウィンドウ関数 wi_string( )は，まず構造体にセットされたメンバの x，y，clm_cnt，ln_cnt の値をチェックします．

なお，ここでは構造体のメンバ名をいちいち書くのが面倒なので，よく使うメンバは，いったんローカル変数に代入しています(とくに意味があってコピーしているわけではない)．

(2) 画面セーブ用のメモリの確保

つぎにポップアップ・ウィンドウのサイズを調べ，gettext( )，puttext( )で使うためのメモリを malloc ( )により確保します．ここで x，y は**図 4** のようにポップアップ・ウィンドウの左上の座標ではなく，内部ウィンドウの開始座標ですので，周囲の枠のぶんだけ余分に確保する必要があります．また，x，y の最低値は枠が書かれるぶんを入れるので，それぞれ「2 以上」ということになります．同じように clm_cnt，ln_cnt もウィンドウ内の広さを指定します(枠の大きさではない)．

確保するメモリ量は，前述のようにウィンドウの大きさの 4 倍です．確保したメモリは gettext( )によってポップアップ・ウィンドウをオープンする前に表示

されているものを保存し，ウィンドウを閉じるときにputtext()で出してやるわけです．

(3) 画面属性のセーブ

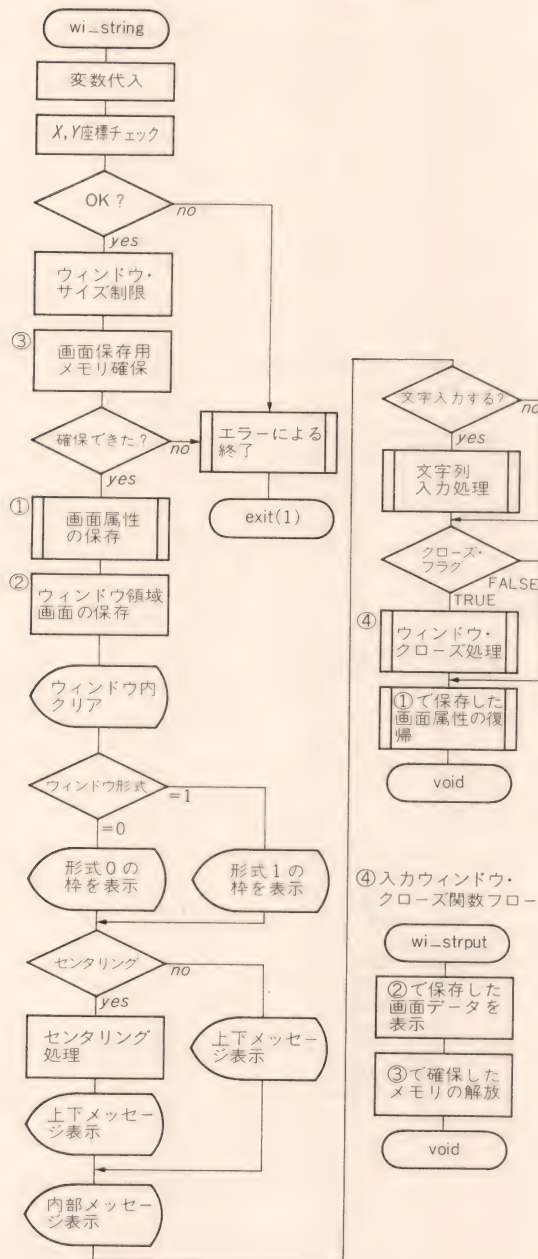
つづいて現在の画面属性をsave_textinfo()関数により保存してから，前述のようにポップアップ・ウィンドウを表示する領域の画面データを確保したメモリに取り込みます．

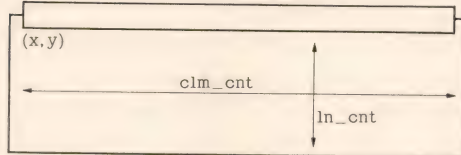
(4) ウィンドウ領域のクリア

〔リスト1〕ウィンドウ作成用構造体の定義など

```
/*
    テキスト画面用ウィンドウ・ライブラリ
    Turboc v2.0     89/6/12    R.GOTOH
*/

#include <stdio.h>
#include <dos.h>
#include <io.h>
#include <conio.h>
#include <fcntl.h>
#include <sys¥stat.h>
#include <string.h>
#include <alloc.h>
#include <stdlib.h>
#include <bios98.h>
#include <ctype.h>
#include <dir.h>
#include <mem.h>

typedef unsigned char   uchar;
typedef unsigned short  ushort;
typedef unsigned long   ulong;
typedef unsigned short  boolean;

#define HIST_LINE   5   /* command history buffer line cnt */
#define HIST_CHAR   35  /* command history buffer char cnt */

/*********************************
    key
*********************************/
#define UP_KEY      0x5     /* ^X */
#define DOWN_KEY    0x18    /* ^E */
#define RIGHT_KEY   0x4     /* ^D */
#define LEFT_KEY    0x13    /* ^S */
#define ESC_KEY     0x1b
#define BS_KEY      0x8
#define CR_KEY      0xd
/*********************************
    etcetra
*********************************/
#define FALSE               0
#define TRUE                1
#define ON                  1
#define OFF                 0
/*********************************
    入力ウィンドウ用構造体
*********************************/
struct _wi {
    ushort  x, y,
            clm_cnt,
            ln_cnt,
            color,
            inn_color,
            style;
    uchar   disp_top[82],
            disp_inn[82],
            disp_btm[82];
    boolean centering;
    uchar   get_buff[256];
    ushort  get_cnt;
    uchar   get_zf,
            get_zt;
    boolean dsw;
    uchar   *tptr;
};
/*********************************
    選択ウィンドウ用構造体
*********************************/
struct _ws {
    ushort  x, y,
            clm_cnt,
            ln_cnt,
            color,
            inn_color,
            sb_color,
            style;
    uchar   disp_top[82],
            disp_btm[82],
            disp_inn[11][82],
            get_string[82],
            get_result;
    boolean centering;
    boolean dsw;
    boolean input;
    uchar   *tptr;
};
/*********************************
        画面保存用構造体
*********************************/
struct _textinfo {
    ushort  mode,x,y,attr,cursor,bank,
            wleft,wtop,wright,wbottom;
};
/*********************************
    Global variable
*********************************/
uchar   Push_key[128];      /* for my_?? key function */
/*********************************
    Prototype
*********************************/
uchar   disp_yn_mess(uchar *str);
uchar   select(void);
uchar   sel_and_in(uchar *disp,
                   uchar hist[][HIST_CHAR], uchar *cmd_str);
void    string_in(struct _wi *wi);
void    wi_select(struct _ws *ws);
void    wi_string(struct _wi *wi);
uchar   wi_input(struct _wi *wi);
void    save_textinfo(struct _textinfo*);
void    load_textinfo(struct _textinfo*);
void    example(void);
void    wi_strput(struct _wi *wi);
void    pabort(uchar*);
void    wi_selbar(struct _ws *ws);
void    wi_iselbar(struct _ws *ws);
void    wi_selput(struct _ws *ws);
ushort  my_kbhit(void);
uchar   my_getch(void);
void    my_ungetch(uchar key);
void    my_ungetstr(uchar *str);
void    save_allfkey(uchar *buff);
void    load_allfkey(uchar *buff);
void    set_fkey(ushort knum, uchar *data);
```

つぎに window( )関数を使って画面領域を決め，その中をクリアします．この window( )関数はクリアする領域を制限するためだけに使用していますが，これによって gotoxy( )関数使用時の座標指定位置が変わってきますので，load_textinfo( )をコールして座標系を元に戻しておきます．

以上で画面属性の保存，画面データの保存，ポップアップ・ウィンドウ内のクリアなどの前準備ができま

〔図5〕ウィンドウ座標(パラメータ x, y の意味)

〔図4〕入力ウィンドウ関数のフロー

〔リスト2〕ウィンドウ作成用関数群①

```
/************************************************************
*                                                           *
*              選択ウィンドウのオープンと、選択             *
*                                                           *
*  struct _ws {                                             *
*      ushort  x, y,             : オープン位置             *
*              clm_cnt,          : カラム数                 *
*              ln_cnt,           : 行数                     *
*              color,            : 外枠の色                 *
*              inn_color,        : メッセージの色           *
*              sb_color,         : セレクトバーの色         *
*              style;            : ウィンドウ形式(0 or 1)   *
*      uchar   disp_top[82],     : TOP に表示する文字列     *
*              disp_btm[82],     : BOTTOM に表示する文字列  *
*              disp_inn[17][82], : 内部に表示する文字列     *
*              get_string[82],   : 入力された文字列         *
*              get_result;       : 選択された番号           *
*      boolean centering;        : centering flag ON/OFF    *
*      boolean dsw;              : TRUE なら画面を元に戻す  *
*      boolean input;            : ON なら文字列を入力する  *
*      uchar   *tptr;            : テキスト保存用バッファ   *
*  };                                                       *
*                                                           *
*  戻り値：void                                             *
*                                                           *
*  ウィンドウを重ねるには、dsw=FALSE で wi_select() を      *
*  枚数分実行後に wi_selput(struct _ws*) をコールする。     *
*                                                           *
************************************************************/
void wi_select(struct _ws *ws)
{
struct  _textinfo t;
ushort  i, x, y, lnc, clmc, barlen, messlen;
uchar   *tptr, bar[82];

    lnc = ws->ln_cnt;
    clmc = ws->clm_cnt;

    x = ws->x;
    y = ws->y;
    if (x<2 || y<2) pabort("X or Y bad position in WI_SELECT");
    if (clmc > 78) clmc = 78;
    if (lnc  > 10) lnc  = 10;
    if (NULL == (tptr=(uchar*)malloc((lnc+2)*(clmc+2)*4)))
        pabort("Malloc in < WI_SELECT >");
    save_textinfo(&t);
    gettext(x-1,y-1,x+clmc,y+lnc,tptr);
    textcolor(ws->color);                 /* open window */
if (ws->style == 0) {                 /* window style 0 */
    textreverse(REVERSE);
    memset(bar, ' ', clmc+2);bar[clmc+2] = '¥0';
    gotoxy(x-1,y-1);cputs(bar); /* write bar top & bottom */
    gotoxy(x-1, y+lnc);cputs(bar);
    for (i=0 ; i<lnc ; i++) {   /* write bar left & right */
        gotoxy(x-1, y+i);cputs(" ");
```

したのでウィンドウの枠を書き始めます.

(5) ウィンドウの枠の表示

枠の形は，ウィンドウ・スタイルを指定するメンバ wi－＞style によって異なります.

MIFES 風のウィンドウを作る場合には，wi－＞style に 0 がセットされています．このタイプの枠は，スペース・コードを反転表示すれば OK です.

wi－＞style == 1の場合(図1のようなウィンドウ)，枠の角や線にグラフィック・コードを使いますので，textmode(BG8025)を使って画面モードをグラフィック有効状態にしてから書いてやります．ここで，もし上下の枠を書くときに枠上に表示するメッセージがあれば(wi－＞disp_top あるいは wi－＞disp_btm に文字列が設定されていれば)スペース・コード反転型のバーを書き，メッセージがなければグラフィック・コードで線を引きます.

〔リスト2〕ウィンドウ作成用関数群②

```
        gotoxy(x+clmc, y+i);cputs(" ");
    }
} else {                            /* window style 1         */
    textmode(BG8025);
    textcolor(ws->color);
    textreverse(NOREVERSE);
    gotoxy(x-1, y-1);      cputs("¥x98");    /* left  top */
    gotoxy(x+clmc, y-1);  cputs("¥x99");    /* right top */
    gotoxy(x-1, y+lnc);   cputs("¥x9A");    /* left  btm */
    gotoxy(x+clmc, y+lnc);cputs("¥x9B");    /* right btm */
    memset(bar, 0x95, clmc);bar[clmc] = '¥0';
    gotoxy(x, y-1);  cputs(bar);            /* top line */
    gotoxy(x, y+lnc);cputs(bar);            /* btm line */
    for (i=0; i<lnc; i++) {
        gotoxy(x-1, y+i);cputs("¥x96");
        gotoxy(x+clmc, y+i);cputs("¥x96");
    }
    textmode(VL8025);
    textcolor(ws->color);

    textreverse(REVERSE);
    memset(bar, ' ', clmc);bar[clmc-2] = '¥0';
    if (ws->disp_top[0]) {
        gotoxy(x+1,y-1);cputs(bar);       /* write bar top */
    }
    if (ws->disp_btm[0]) {
        gotoxy(x+1, y+lnc);cputs(bar);   /* write bar btm */
    }
}

if (ws->centering == ON) {     /* disp. message centering */
    barlen = strlen(bar);
    if (ws->style == 0) barlen -= 4;
    if (ws->disp_top[0]) {          /* disp. top message   */
        messlen = strlen(ws->disp_top);      /* centering */
        gotoxy(x+2+((barlen-messlen)>>1)-1, y-1);
        cputs(ws->disp_top);
    }
    if (ws->disp_btm[0]) {      /* disp. bottom message   */
        messlen = strlen(ws->disp_btm);       /* centering */
        gotoxy(x+2+((barlen-messlen)>>1)-1, y+lnc);
        cputs(ws->disp_btm);
    }
} else {
    gotoxy(x+2, y-1);  cputs(ws->disp_top);
        gotoxy(x+2, y+lnc);cputs(ws->disp_btm);
    }

    if (ws->input == ON) {               /* disp. inner message */
        ws->disp_inn[lnc-1][0] = '¥0';
        textreverse(REVERSE);
    } else textreverse(NOREVERSE);
    memset(bar, ' ', clmc);bar[clmc] = '¥0';
    textcolor(ws->inn_color);
    for (i=0; i<lnc ; i++) {
        if (ws->input == ON) {
            if (i == lnc-1) textreverse(NOREVERSE);
            else textreverse(REVERSE);
        } else textreverse(NOREVERSE);
        gotoxy(x, y+i);cputs(bar);
        gotoxy(x, y+i);
        if (ws->input == OFF) {
            textcolor(ws->color);
            cprintf(" %ld| ",i);
        }
        textcolor(ws->inn_color);
        cprintf("%s",ws->disp_inn[i]);
    }
    if (ws->input == ON) wi_iselbar(ws);
    else wi_selbar(ws);
    ws->tptr = tptr;
    if (ws->dsw==TRUE) wi_selput(ws);
    load_textinfo(&t);
}
/*********************************
*  ウィンドウ・メニューの選択   *
*********************************/
void wi_iselbar(struct _ws *ws)
{
void    ws_istr(struct _ws *ws, uchar rec);
uchar   rec, esc_flag;
short   sel;
ushort  x, y, len;

    textcursor(DISP_CURSOR);
    textreverse(NOREVERSE);
    x = ws->x;
    y = ws->y;

    ws->get_string[0] = '¥0';
    esc_flag = OFF;
    for (sel=ws->ln_cnt-1 ; sel!='¥r'; ) {
        if (sel == ws->ln_cnt-1) {
            len = strlen(ws->get_string);
            gotoxy(x+len, y+sel);
        } else gotoxy(x, y+sel);
        rec = my_getch();
        switch (rec) {
            case 0x1b:
                   esc_flag = ON;
                   break;
            case DOWN_KEY:
                   if (++sel >= ws->ln_cnt) sel = 0;
                   break;
            case UP_KEY:
                   if (--sel < 0) sel = ws->ln_cnt-1;
                   break;
            default:
                   if (sel == ws->ln_cnt-1) ws_istr(ws, rec);
                   break;
        }
        if (esc_flag == ON) break;
```

書き終わったら textmode(VL8025)により元のモードに戻します.

(6) 枠上のメッセージ(ガイダンス)の表示

ここまででポップアップ・ウィンドウの枠表示が終わりました. つぎに枠上下のメッセージを書きます.

メッセージのセンタリングを行うかどうかを wi－>centering フラグによって調べ, ON なら上下枠の真ん中にメッセージを表示しますし, OFF なら反転部分の左端に表示します(まるっきり左端だと美しくないので1文字ぶん空けてあります).

(7) ウィンドウ内部の既定文字列の表示

最後に, ポップアップ・ウィンドウの内部に wi－>disp_inn に設定されている文字列を表示すればウィンドウのでき上りです.

(8) 文字列の入力

ウィンドウができましたので文字列の入力を行いま

```
        if (rec == '¥r') break;
    }
    if ((!ws->get_string[0]) && sel==ws->ln_cnt-1)
        esc_flag = ON;
    if (esc_flag == ON) {
        ws->get_result = ESC_KEY;
        strcpy(ws->get_string, "¥x1b");
    } else {
        ws->get_result = sel;
        if (sel != ws->ln_cnt-1)
            strcpy(ws->get_string, ws->disp_inn[sel]);
    }
}
/*********************************
*  最下行への文字列入力          *
*********************************/
void ws_istr(struct _ws *ws, uchar rec)
{
ushort  i;

    i = strlen(ws->get_string);
    if (rec == ' ') rec = ',';
    if (i > 0) {
        if (rec == BS_KEY) {
            ws->get_string[--i] = '¥0';
            gotoxy(wherex()-1,wherey());
            cputs(" ");
            gotoxy(wherex()-1,wherey());
        }
    }
    if (i < ws->clm_cnt-1) {
        if ((rec>=' ') && (rec<='z') && (rec!=BS_KEY)) {
            ws->get_string[i++] = rec;
            ws->get_string[i] = '¥0';
            cprintf("%c",rec);
        }
    }
}
/*********************************
*  ウィンドウ・メニューの選択    *
*********************************/
void wi_selbar(struct _ws *ws)
{
uchar   rec, esc_flag;
short   sel, selc;
uchar   bar[82];
ushort  clmc, x, y;
    textcursor(NODISP_CURSOR);
    clmc = ws->clm_cnt;
    x = ws->x;
    y = ws->y;
    memset(bar, ' ', clmc-2);
    bar[clmc-2] = '¥0';
    esc_flag = OFF;
    for (selc=sel=0 ; sel!='¥r'; ) {
        textreverse(REVERSE);
        textcolor(ws->sb_color);
        gotoxy(x+1, y+sel);cputs(bar);
        gotoxy(x+1, y+sel);
        cprintf("%1d| ",sel);cprintf("%s",ws->disp_inn[sel]);
        rec = my_getch();
        switch (rec) {
            case 0x1b:      esc_flag = ON;
                            break;
            case DOWN_KEY:  selc = sel;
                            if (++sel >= ws->ln_cnt) sel = 0;
                            break;
            case UP_KEY:    selc = sel;
                            if (--sel < 0) sel = ws->ln_cnt-1;
                            break;
            default:        break;
        }
        textreverse(NOREVERSE);
        gotoxy(x+1, y+selc);cputs(bar);
        textcolor(ws->color);gotoxy(x+1, y+selc);
        cprintf("%1d| ",selc);
        textcolor(ws->inn_color);
        cprintf("%s",ws->disp_inn[selc]);

        if (esc_flag == ON) break;
        if (rec == '¥r') break;
        if (rec>='0' && rec-'0'<ws->ln_cnt) {
            sel = rec - '0';
            break;
        }
    }
    if (esc_flag == ON) ws->get_result = rec;
    else ws->get_result = sel;
}
/*********************************
*  画面を元に戻す                *
*********************************/
void wi_selput(struct _ws *ws)
{
    puttext(ws->x-1,ws->y-1,ws->x+ws->clm_cnt,
                        ws->y+ws->ln_cnt,ws->tptr);
    free(ws->tptr);
}

/************************************************************
*                                                           *
*          文字列入力ウィンドウのオープンと、キー入力       *
*                                                           *
*          キー入力しない場合は、「get_cnt = 0」 とする。   *
*                                                           *
*  struct _wi {                                             *
*      ushort  x, y,           : オープン位置               *
*              clm_cnt,        : カラム数                   *
*              ln_cnt,         : 行数                       *
*              color,          : 外枠の色                   *
```

す．ただし wi－＞get_cnt メンバを調べて 0 の場合は入力ルーチンを呼びません．

入力ルーチンでは ESC キーか RET キーが押されるまでキー・データを取り込みます．少しゴチャゴチャしているのは，2 行以上にわたって入力した場合にも BS キーでちゃんと消せるようにするためです．window 関数を使ってウィンドウを作り，キー入力させると 2 行以上にわたったときに BS キーでは現在の行しか消せません(上の行には戻らない)ので，このような処理になっています．

入力されたキー・データは wi－＞get_buff［］に納められますが，get_buff 配列以上の文字数が入力された場合エラーとなりますので，wi－＞get_cnt には必ず get_buff 配列以下の値を設定しておかねばなりません．

なお，キー入力中に ESC キーが押された場合には

〔リスト 2〕ウィンドウ作成用関数群 ③

```
*               inn_color,      : 内枠の色                    *
*               style;          : ウィンドウ形式(0 or 1)      *
*       uchar   disp_top[82],   : TOP に表示する文字列        *
*               disp_inn[82],   : 内部に表示する文字列        *
*               disp_btm[82];   : BOTTOM に表示する文字列     *
*       boolean centering;      : centering flag ON/OFF       *
*       uchar   get_buff[256];  : キー入力バッファ            *
*       ushort  get_cnt;        : キー入力文字数              *
*       uchar   get_zf,         : キー入力範囲 (from)         *
*               get_zt;         : キー入力 ( to )             *
*       boolean dsw;            : TRUE なら画面を元に戻す     *
*       uchar   *tptr;          : テキスト保存用バッファ      *
*   };                                                        *
*                                                             *
*   戻り値：void                                              *
*                                                             *
*   ウィンドウを重ねるには、dsw=FALSE にて wi_string() を     *
*   枚数ぶん実行後に wi_strput(struct _wi*) をコールする。    *
*                                                             *
***************************************************************/
void wi_string(struct _wi *wi)
{
struct  _textinfo t;
ushort  i, x, y, lnc, clmc, barlen, messlen;
uchar   *tptr, bar[82];

    lnc = wi->ln_cnt;
    clmc = wi->clm_cnt;
    x = wi->x;
    y = wi->y;
    if (x<2 || y<2) pabort("X or Y bad position in WI_STRING");
    if (clmc > 78) clmc = 78;
    if (lnc  > 23) lnc  = 23;
    if (NULL == (tptr=(uchar*)malloc((lnc+2)*(clmc+2)*4)))
        pabort("Malloc in < WI_STRING >");
    save_textinfo(&t);
    gettext(x-1,y-1,x+clmc,y+lnc,tptr);
    window(x-1,y-1,x+clmc,y+lnc);               /* clear window */
    clrscr();
    load_textinfo(&t);

    textcursor(NODISP_CURSOR);                  /* open window */
    if (wi->style == 0) {                       /* window style 0 */
        textcolor(wi->color);
        textreverse(REVERSE);
        memset(bar, ' ', clmc+2);bar[clmc+2] = '¥0';
        gotoxy(x-1,y-1);cputs(bar); /* write bar top & bottom */
        gotoxy(x-1,y+lnc);cputs(bar);
        for (i=0 ; i<lnc ; i++) {   /* write bar left & right */
            gotoxy(x-1, y+i);cputs(" ");
            gotoxy(x+clmc, y+i);cputs(" ");
        }
    } else {                                    /* window style 1 */
        textmode(BG8025);
        textcolor(wi->color);                   /* open window */
        textreverse(NOREVERSE);
        gotoxy(x-1, y-1);     cputs("¥x98");    /* left  top */
        gotoxy(x+clmc, y-1);  cputs("¥x99");    /* right top */
        gotoxy(x-1, y+lnc);   cputs("¥x9A");    /* left  btm */
        gotoxy(x+clmc, y+lnc);cputs("¥x9B");    /* right btm */
        memset(bar, 0x95, clmc);bar[clmc] = '¥0';
        gotoxy(x, y-1);  cputs(bar);            /* top line */
        gotoxy(x, y+lnc);cputs(bar);            /* btm line */
        for (i=0; i<lnc; i++) {
            gotoxy(x-1, y+i);cputs("¥x96");
            gotoxy(x+clmc, y+i);cputs("¥x96");
        }
        textmode(VL8025);
        textcolor(wi->color);

        textreverse(REVERSE);
        memset(bar, ' ', clmc);bar[clmc-2] = '¥0';
        if (wi->disp_top[0]) {
            gotoxy(x+1,y-1);cputs(bar);    /* write bar top */
        }
        if (wi->disp_btm[0]) {
            gotoxy(x+1, y+lnc);cputs(bar); /* write bar btm */
        }
    }

    if (wi->centering == ON) {   /* disp. message centering */
        barlen = strlen(bar);
        if (wi->style == 0) barlen -= 4;
            if (wi->disp_top[0]) {       /* disp. top message   */
                messlen = strlen(wi->disp_top);    /* centering */
                gotoxy(x+2+((barlen-messlen)>>1)-1, y-1);
                cputs(wi->disp_top);
            }
            if (wi->disp_btm[0]) {    /* disp. bottom message   */
                messlen = strlen(wi->disp_btm);    /* centering */
                gotoxy(x+2+((barlen-messlen)>>1)-1, y+lnc);
                cputs(wi->disp_btm);
            }
    } else {
        gotoxy(x+2, y-1);  cputs(wi->disp_top);
        gotoxy(x+2, y+lnc);cputs(wi->disp_btm);
    }
    textreverse(NOREVERSE);                     /*  inner message */
    textcolor(wi->inn_color);
    gotoxy(x, y);cputs(wi->disp_inn);

    if (wi->get_cnt) {
        if (ESC_KEY == wi_input(wi)) wi->get_buff[0] = ESC_KEY;
    }
    wi->tptr = tptr;
    if (wi->dsw == TRUE) wi_strput(wi);
    load_textinfo(&t);
}
/*********************************
*  ウィンドウへの文字列の入力  *
*   return:if ESC -> 0x1b      *
*********************************/
uchar wi_input(struct _wi *wi)
```

wi_get_buff [0] に ESC キー・コード(0x1B)をセットしています.

(9) ウィンドウのクローズおよび呼出し側への戻り

キー入力処理が終わったらウィンドウを書く前に保存しておいた画面を表示するのですが, もし wi－>dsw メンバが OFF であった場合は, ウィンドウをオープンしたままで呼出し側に戻します. これはウィンドウを何枚か重ねる場合に使います. wi－>tptr には gettext 関数で保存した元の画面の情報を保存するためのポインタ(最初に malloc で確保したもの)が入っていますので, 呼出し側でウィンドウを閉じるには wi 構造体を引数として wi_strput 関数をコールするだけですみます. ただしウィンドウを何枚かオーバラップ(重ね書き)した場合には, スタックと同じように後から書いたものほど先に消すようにしなければなりません.

```
{
uchar   rec;
ushort   x, y, i;

    if (sizeof(wi->get_buff) <= wi->get_cnt)
                            pabort("BAD get_cnt in WI_INPUT");
    memset(wi->get_buff, 0, wi->get_cnt+1);
    textcursor(DISP_CURSOR);
    x = wherex();
    y = wherey();
    for (i=0; (rec=my_getch())!='¥r' && rec!=ESC_KEY; ) {
        if (i > 0) {
            if (rec == BS_KEY) {
                wi->get_buff[--i]='¥0';
                if (wherex()-1 < x)
                   gotoxy(wi->x+wi->clm_cnt-1, --y);
                else gotoxy(wherex()-1,wherey());
                cputs(" ");
                if (wherex()-1 < x)
                   gotoxy(wi->x+wi->clm_cnt-1, --y);
                gotoxy(wherex()-1,wherey());
            }
        }
        if (i < wi->get_cnt) {
            if ((rec>=wi->get_zf) && (rec<=wi->get_zt) &&
                                     (rec!=BS_KEY)) {
                wi->get_buff[i++]=rec;
                cprintf("%c",rec);
                if (wherex() == wi->x+wi->clm_cnt)
                   gotoxy(x,++y);
            }
        }
    }
    return(rec);
}
/********************************
*  画面を元に戻す              *
********************************/
void wi_strput(struct _wi *wi)
{
    puttext(wi->x-1,wi->y-1,wi->x+wi->clm_cnt,
                        wi->y+wi->ln_cnt,wi->tptr);
    free(wi->tptr);
}
/********************************
    画面属性の保存と復帰
********************************/
void save_textinfo(struct _textinfo *t)
{
struct  text_info tinf;

    gettextinfo(&tinf);
    t->mode = tinf.currmode;
    t->x = tinf.curx[tinf.bank];
    t->y = tinf.cury[tinf.bank];
    t->attr = tinf.attribute[tinf.bank];
    t->cursor = tinf.cursor[tinf.bank];
    t->bank = tinf.bank;
    t->wleft = tinf.winleft[tinf.bank];
    t->wtop = tinf.wintop[tinf.bank];
    t->wright = tinf.winright[tinf.bank];
    t->wbottom = tinf.winbottom[tinf.bank];
}
void load_textinfo(struct _textinfo *t)
{
    textmode(t->mode);
    textbank(t->bank);
    window(t->wleft, t->wtop, t->wright, t->wbottom);
    gotoxy(t->x,t->y);
    textattr(t->attr);
    textcursor(t->cursor);
}
/********************************
    Fキーの保存、復帰、設定
********************************/
void save_allfkey(char *s)
{
short   ds;

    ds = _DS;
    _DS = FP_SEG(s);
    _AX = 0;
    _DX = (unsigned int)s;
    _CL = 0x0c;
    geninterrupt(0xdc);
    _DS = ds;
}
void load_allfkey(char *s)
{
short   ds;

    ds = _DS;
    _DS = FP_SEG(s);
    _AX = 0;
    _DX = (unsigned int)s;
    _CL = 0x0d;
    geninterrupt(0xdc);
    _DS = ds;
}
void set_fkey(ushort n, uchar *s)
{
short   ds;

    ds = _DS;
    _DS = FP_SEG(s);
    _CL = 0xd;
    _AX = n;
    _DX = (short)s;
    geninterrupt(0xdc);
    _DS = ds;
}
```

以上でウィンドウ処理が終わりましたので，このウィンドウ・ルーチンへくる前の状態に画面属性を戻します．

メニュー選択用ウィンドウ関数 wi_select についても図6に処理の流れを示しておきます．

▶キー入力関係の関数(my_getch( ))

ところでキー入力ルーチンで my_getch( ) という関数を使っていますが，これらは，キー入力関係の関数がそのままでは不便なところがあったので，いくつか作ったものです(図7参照)．

● my_kbhit( )

my_kbhit( ) は kbhit( ) の改造版です．kbhit( ) は

〔リスト2〕ウィンドウ作成用関数群④

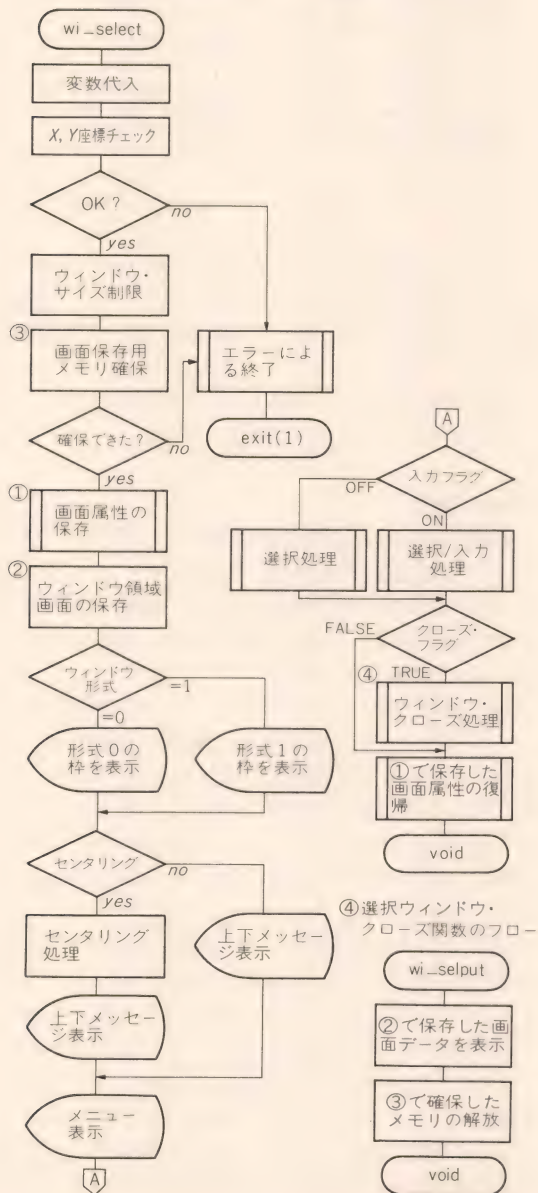
```
/*********************************
         キー入力関係
*********************************/
uchar my_getch(void)
{
uchar    my_getchg(uchar ah);
uchar    al, ah, key;
    if (Push_key[0]) {  /* if ungetch ... */
        key = Push_key[0];
        movmem(&Push_key[1], &Push_key[0], strlen(Push_key));
    } else {
        _AH = 0x0;  → 疑似レジヘタへの代入
        geninterrupt(0x18); → 単純に INT 18H に展開される
        al = _AL;
        ah = _AH;
        key = al ? al: my_getchg(ah);
    }
    return(key);
}
uchar my_getchg(uchar ah)
{
uchar     key;
    if (ah == 0x3a) key = UP_KEY;  /* arrow  */
    else if (ah == 0x3d) key = DOWN_KEY;
    return (key);
}
void my_ungetch(uchar key)
{
ushort  len;
    len = strlen(Push_key);
    Push_key[len] = key;
    Push_key[len+1] = '¥0';
    if (strlen(Push_key) == sizeof(Push_key))
        pabort("Can't ungetch");
}
void my_ungetstr(uchar *str)
{
    strcat(Push_key, str);
    if (strlen(Push_key) == sizeof(Push_key))
        pabort("Can't ungetch");
}
ushort my_kbhit(void)
{
    if (Push_key[0]) return (strlen(Push_key));
    return (peek(0,0x528));
}
/*********************************
           その他
*********************************/
void pabort(uchar *string)
{
    printf("¥n¥r¥n¥rABORT : %s",string);
    exit(1);
}
```

〔リスト3〕「かまし関数」の例

```
/*****************************************************
    「Y／N」選択ウィンドウのオープン
    return : input key data
*****************************************************/
uchar disp_yn_mess(uchar *str)
{
struct  _wi wi;
ushort  len;
uchar   key;

    len = strlen(str);            /* メッセージの長さに応じて */
    if (len >= 72) str[72] = '¥0';  /* ウィンドウの幅をかえる */
    len = len < 18 ? len = 18: len+2 ;
    wi.clm_cnt = len;               /* ウィンドウの大きさ      */
    wi.ln_cnt = 1;
    wi.x = 20;                        /* X=20, Y=10 に表示 */
    wi.y = 10;
    wi.color = T_RED;                 /* 外枠の色を設定     */
    wi.inn_color = T_LIGHTCYAN;       /* メニューの色       */
    strcpy(wi.disp_top, "Message window");/* 上枠のメッセージ */
    strcpy(wi.disp_inn, str);       /* 表示する文字列をコピー */
    strcpy(wi.disp_btm, "Yes / No");/* 下枠に「Y／N」を表示 */
    wi.centering = ON;              /* センタリングする        */
    wi.get_cnt = 0;                 /* 文字列は入力しない      */
    wi.dsw = FALSE;               /* 入力後ウィンドウ閉じない */
    wi.style = 0;                    /* ウィンドウ形式=0      */
    wi_string(&wi);                  /* ウィンドウ・オープン   */
    do {                                    /* キー入力待ち   */
        key = toupper(my_getch());
    } while (key!='Y' && key!='N');
    wi_strput(&wi);                  /* ウィンドウを閉じる     */
    return (key);                    /* 入力したキーを返す     */
}
/*****************************************************
    選択ウィンドウのオープン
    return : selected number or ESC_KEY
*****************************************************/
uchar select(void)
{
struct  _ws ws;

    ws.x = 15;                        /* X=15, Y=10 に表示     */
    ws.y = 10;
    ws.clm_cnt = 50;                  /* ウィンドウの大きさ   */
    ws.ln_cnt = 3;
    ws.color = T_GREEN;               /* 外枠の色を設定        */
    ws.inn_color = T_YELLOW;          /* メニューの色          */
    ws.sb_color = T_WHITE;            /* 選択バーの色          */
    strcpy(ws.disp_top, "選択して下さい");/* 上枠のメッセージ */
    strcpy(ws.disp_inn[0], "Select menu 1");
                                      /* 表示させるメニュー */
    strcpy(ws.disp_inn[1], "Select menu 2");
    strcpy(ws.disp_inn[2], "Select menu 3");
    strcpy(ws.disp_btm, "");          /* 下枠はメッセージ無し */
    ws.centering = ON;                /* センタリングする      */
    ws.dsw = TRUE;                    /* ウィンドウを重ねない */
    ws.style = 1;                     /* ウィンドウ形式=1     */
    ws.input = OFF;                   /* 文字列入力はしない   */
    wi_select(&ws);                   /* ウィンドウ・オープン */
    return (ws.get_result);           /* 選択された番号を返す */
}
```

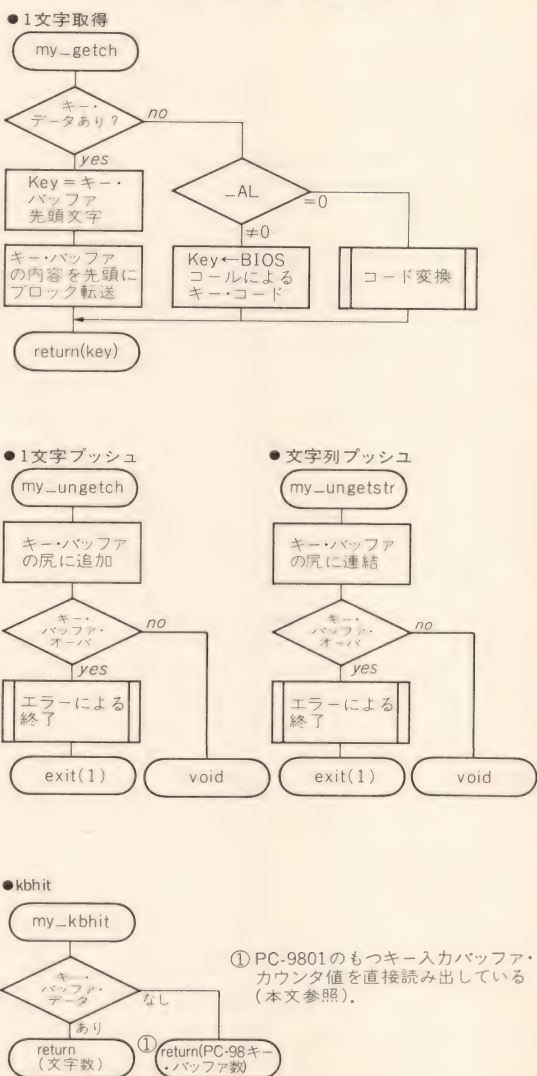
^Cを入力するとDOSに戻ってしまったり，画面に^Cの文字が出たりします．これを防ぐにはbios98key()などを使う方法もあるのですが，キーが押されたかどうかを調べるだけのために，わざわざこんな大きな関数を呼ぶのも芸がないので，セグメント0の環状キー・バッファの文字カウンタをpeekして調べています．Turbo Cでは，dos.hをインクルードしておくとpeekやpokeなどの命令がインラインに展開され最高のスピードを得ることができますし，^Cも恐くありません．

● my_getch()

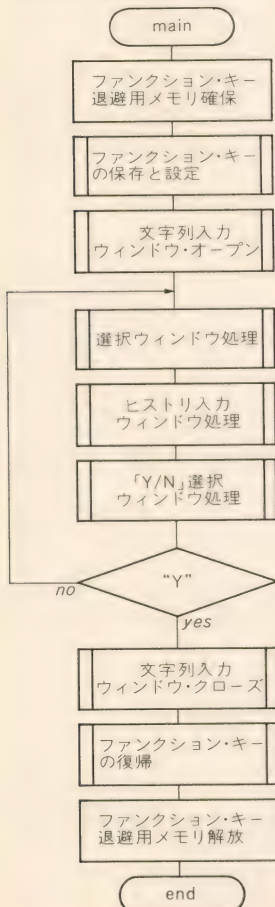
〔図6〕選択ウィンドウ関数のフロー

my_getch()はgetch()の代わりです．getch()は^Cは取れるのですが，グラフィック・コードはブザーが鳴って取り込めません．また，ungetch()で1文字しかプッシュ・バックできないのも不便だったので，my_ungetch()，my_ungetstr()と組み合わせて使うようにしてあります．

my_getch()では，PC-9801のROM-BIOSをコールしています．これでグラフィック・キーも取り込めます．また，もしmy_ungetch()やmy_ungetstr()で文字列がプッシュされている場合には，そこからもってきます．ところでこれらプッシュ用のバッファはリング・バッファとすべきなのでしょうが，リング・ポインタの管理にグローバル変数を使わなければならず，あまりグローバル変数を増やしたくなかったし速度的

〔図7〕キー入力関係の関数のフロー

〔リスト4〕サンプル・プログラム

```
/*****************************************************
                    main example
*****************************************************/
void main(void)
{
struct  _wi a;
static  uchar   cmd_hist[HIST_LINE][HIST_CHAR],
                cmd_str[82];
uchar   *fkey;

    if (NULL == (fkey=(uchar*)malloc(400)))
                                    pabort("malloc in main");
    save_allfkey(fkey);     /* 全ファンクション・キーの保存 */
    set_fkey(25, "¥x05");   /* UP arrow key   */
    set_fkey(28, "¥x18");   /* DOWN arrow key */

    string_in(&a);          /* 文字列入力ウィンドウのオープン */
    for (;;) {
        select();
        sel_and_in("データ入力", cmd_hist, cmd_str);
        if ('Y' ==
           disp_yn_mess("サンプルを終了して良いですか  ?"))
          break;
    }
    wi_strput(&a);       /* 文字列入力ウィンドウのクローズ */

    load_allfkey(fkey);  /* 全ファンクションキーを戻す */
    free(fkey);
}
/*****************************************************
    文字列入力ウィンドウのオープン
    return :nothing
*****************************************************/
void string_in(struct _wi *wi)
{
    wi->clm_cnt = 75;    /* ウィンドウの大きさ       */
    wi->ln_cnt = 15;
    wi->x = 2;           /* X=2, Y=5 に表示          */
    wi->y = 5;
    wi->color = T_YELLOW;       /* 外枠の色を設定    */
    wi->inn_color = T_WHITE;    /* メニューの色を設定 */
    strcpy(wi->disp_top, "Input window");
                                /* 上枠のメッセージ  */
    strcpy(wi->disp_inn, "please>");
                                /* ウィンドウ内文字列 */
    strcpy(wi->disp_btm, " "); /* 下枠を付ける       */
    wi->centering = ON;         /* センタリングする  */
    wi->get_cnt = 100;          /* 100 文字入力する  */
```

〔図8〕main関数のフロー

main
ファンクション・キー退避用メモリ確保
ファンクション・キーの保存と設定
文字列入力ウィンドウ・オープン
選択ウィンドウ処理
ヒストリ入力ウィンドウ処理
「Y/N」選択ウィンドウ処理
"Y"
no
yes
文字列入力ウィンドウ・クローズ
ファンクション・キーの復帰
ファンクション・キー退避用メモリ解放
end

にもキー入力時はそれほど問題とならないので，単純なブロック転送を行っています(というよりも，実際には後から付け加えたため，ただ面倒くさかっただけです).

なお，BIOSコールで疑似レジスタを使っていますが，これについては次号で補足します.

▶ main関数

リスト4のmain関数は，今回のポップアップ・ウィンドウ作成関数群を利用したサンプル・プログラムです．とくに意味のある処理を行っているわけではありません(**図8**参照)．かまし関数として，string_in，sel_and_inを使用しています.

main関数内では，ファンクション・キー関係の処理(ファンクション・キーの待避，復活，設定)を行っています．初めにユーザの設定したファンクション・キーをsave_allfkey( )で待避しておき，プログラム内で使うファンクション・キー(↑，↓)をset_fkey( )により設定します．プログラム終了時には，load_allfkey( )を使って戻します.

BIOSコール時はセグメントも設定していますので，スモール・モデル以外でも正常に動きます(ときどきセグメントの設定をしていない例があるので念のため).

ところでTurbo CのHugeモデルにはバグがあるようです．先日グローバル変数の構造体をアクセスする関数で，どうしてもうまく動かず，TD(ターボ・デバッガ)で追いかけたら，とんでもない動作をしているのがありました．今のところHugeモデルは使わない

```
    wi->get_zf = 'A';  /* 入力できるのは  A～zまで */
    wi->get_zt = 'z';
    wi->dsw = FALSE;      /* 入力後ウィンドウ閉じない */
    wi->style = 1;            /* ウィンドウ形式＝1 */
    wi_string(wi);           /* ウィンドウ・オープン */
}
/*****************************************************
    ヒストリ機能付き入力ウィンドウのオープン
    return :0 or ESC_KEY
*****************************************************/
uchar sel_and_in(uchar *disp,
        uchar hist[][HIST_CHAR], uchar *string)
{
struct  _ws ws;
ushort  i, hist_cnt;

    for (i=0; i<HIST_LINE; ++i) {
        if (!hist[i][0]) break;
        strcpy(ws.disp_inn[i], hist[i]);
    }
    hist_cnt = i;

    ws.x = 40;                      /* X=40, Y=9 に表示  */
    ws.y = 9;
    ws.ln_cnt = hist_cnt + 1;
    ws.clm_cnt = HIST_CHAR;
    ws.color = T_BLUE;
    ws.inn_color = T_WHITE;
    ws.dsw = TRUE;    /* 入力が終わったらウィンドウを閉じる   */
    ws.input = ON;              /* 文字入力を行う              */
    ws.style = 1;               /* ウィンドウ形式＝1            */
    ws.centering = ON;          /* センタリングする             */
    strcpy(ws.disp_top, disp); /* 上枠にガイダンスを表示する  */
    strcpy(ws.disp_btm,"");    /* 下枠はなし                  */
    wi_select(&ws);
    if (ws.get_result != 0x1b) {
                        /* 入力データをヒストリーバッファに移す */
        if (hist_cnt == HIST_LINE) {
            for (i=0; i<HIST_LINE-1; ++i) strcpy(hist[i], hist[i+1]);
            strcpy(hist[i], ws.get_string);
        } else {
            strcpy(hist[hist_cnt], ws.get_string);
        }
        strcpy(string, ws.get_string);
        return (0);
    }
    return (ESC_KEY);     /* ESC_KEY  入力時 */
}
```

ようにしています．

▶使用上の注意

間違いなく暴走するのは，オープンしていないウィンドウを wi_strput や wi_selput でクローズした場合です．クローズするときに画面保存用のメモリ tptr を free 関数で解放していますので，tptr メンバに正常な値がセットされていないと，とんでもないことになります(tptr メンバへの値の設定はウィンドウ・オープン時に行われるので)．

逆にクローズしないで何回もオープンをつづけた場合は，いずれ画面保存用メモリが不足しますので，エラー原因を表示して DOS へ戻ります．

ウィンドウ用構造体への値の設定ですが，_wi や _ws などの構造体メンバの順番を変えて static で宣言して初期化すれば，リストのように毎回メンバに代入する必要はなくなります．

おわりに

お互いに似たような関数を作っていることと思いますが，それらをどんどん公表して二度手間にならないですむようにしませんか．インターフェース LSI とのやりとりやメニュー・ルーチンなど，読者の皆さんもお作りのことと思います．なるべく便利なものは出し合って，利用できるものは利用するプログラムのリサイクルなんてどうでしょうか．

参考文献

1）『PC-Techknow9800』，㈱システム・ソフト

ごとう・りゅうじ　㈱アドテック システム サイエンス

SOFTWARE NOTE

IBM PC ソフトを PC-9801 で動かす ポーティング手法入門

Igal Blumenreich

米国では，IBM PC(/XT, /AT など)がパソコンの標準機となっている．一方，日本では，PC-9801 が実質的な標準機といえる．両機に互換性はないが，IBM PC 用に開発された膨大な各種ソフト(OS, コンパイラ，ツール，アプリケーションなど)を PC-9801 用に，という需要はひじょうに多い．ここでは同じ 86 系 CPU を用いながら IBM PC から PC-9801 への移植(ポーティングという)が難しいといわれる理由を示しながら，実際のポーティング作業の基本手法をいくつか示していく．移植作業に従事する人だけでなく，パソコンのメカニズムを知るうえでも参考になろう．　(編集部)

本稿では，パソコンのアプリケーション・ソフトウェアの異機種間の移植作業について考えていきたいと思います．そのうちとくに，米国の IBM PC 用ソフトを日本の PC-9801 用ソフトに移植するときの方法についてみていくことにしましょう．

移植せずとも動くソフト

▶ IBM PC と PC-9801 で動くソフト

両者とも，CPU に 86 系のものを用い，MS-DOS を基本 OS として使用します．ですから，基本的な機能，たとえばディスク・ファイルのアクセス，文字の単純な表示などの DOS 内の基本機能ルーチンは，アプリケーション・ソフトからでも MS-DOS 本体部の機能を通して動作するので，なんの移植作業もせずに両機で使用することができるのです．たとえば，つぎのクラシックな C 言語の例文をみてみましょう．

```
main()
{
    printf( "hello, world¥n" );
}
```

これを，IBM PC と PC-9801 の両方をサポートする C コンパイラでコンパイルすると，画面に表示される結果は，いずれも

```
hello, world
```

になります．

▶ どこが違うか

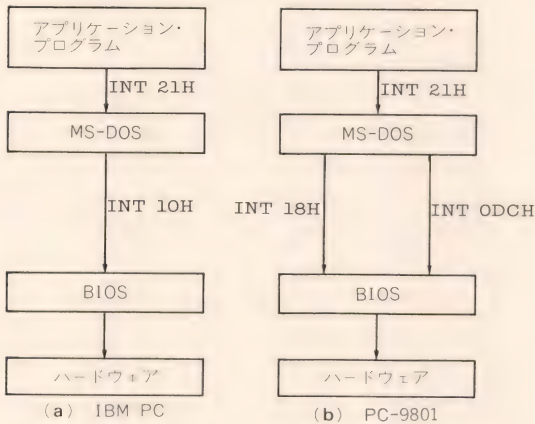
100 %の互換性があるマシンであれば，思考(thinking)も働きも同じなので，当然，結果も同じになります．しかし，IBM PC と PC-9801 の両機種は，互換性が考慮されていない機種同士なので，当然，思考や働きは違ってきます．

ところで，同じ CPU をもつならばコンピュータとして同じ脳をもっているはずです．コンピュータで実行する命令は，いずれにしても CPU で行うわけですから，同じ CPU であれば，理解と実行の方法はただ一つで同じはずです．だとすると，同じ CPU をもっている IBM PC と PC-9801 は，なぜ互換性がないのでしょうか．それは，画面，キーボード，ディスクなどの CPU の周辺にある装置が違うからです．

さらにもう一つ，CPU と周辺装置とのコミュニケーション方法も違います．コンピュータには，周辺装置とのコミュニケーションをとるために，入力と出力の基本システムが用意されています．このシステムを通常 BIOS(Basic Input Output System)とよびます．周辺装置からみると，BIOS の上にオペレーティング・システムがのっています．

IBM PC と PC-9801 のソフトは，ほとんど MS-DOS 上で動いています．MS-DOS は，BIOS を使用して，マシンとユーザとの間を結んでいます．マシンの立上げと同時に，MS-DOS 用のいろいろなサブルーチンはメモリ内に常駐されます．そして，MS-DOS が働く際には，これらのサブルーチンを使用し，キーボードからの入力を読み込んだり，画面上の文字表示，ファイルからの読み書きなどをします．

コンパイルするときは，ソース・ファイルの命令をマシン語になおし，入出力操作をするためにさきほどのサブルーチンをよく使用します．このように，MS-

〔図1〕一般的な BIOS へのアクセス法

DOS のサブルーチンを呼び出すことを通常，**システム・コール**とか **DOS コール**とよびます(INT 21H).

プログラム実行時にシステム・コールがあった場合には，サブルーチンにいって，さらにそこから BIOS のほうに進み結果を得ます(図1).

▶ BIOS の呼出し方

さきほどのCの例題を使ってもう少し具体的に解説しましょう．コンパイラは，

```
printf( "hello, world n" )
```

というステートメントを受けると，CPU はつぎのコマンド・シーケンスとして翻訳します．

```
MOV    AX,  009EH
PUSH   AX
CALL   _PRINTF
```

_PRINTF の部分は，通常，コンパイラから出てくるので，普通のプログラマにはなんら気にする必要はありません．_PRINTF の部分は図2に示します．

画面に文字を表示するためにシステム・コールを行うと，DOS は BIOS を使います．普通，IBM PC 上のプログラムは，INT 10H を使って BIOS を呼び出します(BIOS コール)．一方 PC-9801 では，INT 18H を用いて BIOS をコールします．さらに，DOS 自体がうまく移植されたものであれば，システム・コール INT 21H は BIOS コールを使って(IBM PC の場合 INT 10H，PC-9801 の場合 INT 18H あるいは INT ODCH)，その機能を実現します．

先に示した“hello, world”の例文で，両機とも同じ動作をするという理由は，この INT 21H を使用していることにあります．

〔図2〕_PRINTF ルーチンの概要

```
_PRINTF:
      PUSH    BP
      MOV     BP, SP
      .............
      .............  <-------
      .............          |
      CALL    _FPUTC         |
      .............          |
      .............  --------
      .............

_FPUTC:
      .............
      .............
      INT     21H       ; One of several DOS calls.
      .............
      .............
      CALL    __WRITE   ; Write character/s to consol.
      .............
      .............

__WRITE:
      .............
      .............
      MOV     AX, 40H     ; DOS call function 40H:
                          ; Write one character to consol.
      MOV     BX, 04[BP]  ; File handle.
      MOV     CX, 08[BP]  ; Bytes to write.
      MOV     CX, 06[BP]  ; DS:DX - Buffer
                          ; storing characters to write.
      INT     21H         ; <======= Call MS-DOS
      .............
      .............
```

▶移植が必要となるわけ

しかしながら，現在の開発されているソフトの大部分は，速いレスポンス(反応)と，よりよいパフォーマンスを得るために，いろいろと複雑なテクニックを使用しています．とくに，DOS を介さずに画面に直接アクセスしたり，キーボードを直接制御するアセンブリ・ルーチンを使用します．このようにアプリケーション・ソフトで高速化のテクニックを使うと，IBM PC と PC-9801 は，ハードウェアの違いもあり，BIOS の構成も違うために，互換性のあるソフトからはどんどん遠ざかってしまいます．

たとえば，INT 10H のようなコマンドをコンパイラを通さず直接使用する IBM PC のアプリケーション・ソフトを，PC-9801 で動かそうとすると，コンパイル時にその INT 10H はそのまま残ってしまいます．このため PC-9801 の BIOS とまったく合わなくなり，システムが停止してしまったり，最悪の場合にはハード・ディスクのデータを壊したりハードウェア自体にダメージを与えてしまう危険性があります．

このような理由から，IBM PC 用から PC-9801 用へといったソフトウェアのポーティング(移植)作業は，ひじょうにめんどくさいものとなります．両機のハードウェアに関する深い知識だけでなく，コンパイラやアセンブラなどの各種手法などに精通してなくてはなりません．

移植者は，ポーティングの作業をするさい，できるだけ修正箇所を少なくしなくてはなりません．これは，もちろんオリジナルに近いかたちで再現するために必要なことですし，デバッグのためにも重要です．また，将来のバージョンアップのための考慮もしなくてはなりません．

▶移植の問題点

移植時の多くの問題点は，前述のように入出力関係にあります．PC-9801 と IBM PC との重要な相違点は，

① 基礎 BIOS コール

② 割込み番号

③ バッファのアドレス

④ I/O ポート

などにあります．

移植者は，こういった違いをどのようにして克服するのでしょうか．以下に，そういったポーティングの基本手法のいくつかを紹介しましょう．

ハードウェアの違いに対処する

▶キーボード ―― IBM PC の ALT キーをどうするか

誰もが最初に気付く IBM PC と PC-9801 との違いは，キーボードでしょう．ASCII(アルファベット)の部分は同じですが，そのまわりのキーが少し違います．

まず，ファンクション・キーの数が違います．PC-9801 が 10 個(最近のタイプは 15 個)であるのに対して，IBM PC は 12 個です(ただし，ほとんどのアプリケーションでは 10 個しか使用しない)．

より重要な違いは，IBM PC 側には ALT キーがあるということです．このキーに対応するものは PC-9801 にはありません．逆に，PC-9801 にはカナ・キー，GRPH キー，XFER キー，NFER キー(昔の機種にはない)があり，これは IBM PC にはありません．

IBM PC ソフトを PC-9801 用にポーティングするとき，ALT キーはどうすればよいでしょうか．まず思いつくのが PC-9801 にしかないキーに割り当てることでしょう．しかしカナ・キーは，ALT キーと違ってロック/アンロックのようになっています．つまり IBM PC 側のアプリケーションの多くは，ALT キーを押しながら違うキーを押して起動するようになっています．このように ALT キーは，もう一つのキーとセットになって使用するのです．そして，その押した二つのキーから手を離すとつぎの入力作業ができます(ALT キーはロックされない)．

PC-9801 の GRPH キーは，ALT キーのようにプレス/アンプレスになっています．たとえば Turbo C の日本語版では，Turbo C 環境(TC)の実行時に ALT キーの代わりに GRPH キーを使っています．IBM PC の ALT キーは，左右両側にあるので，PC-9801 の右側にある XFER キーを使うという方法もあります．XFER キーも ALT キーと同じくプレス/アンプレスになっていますが，他の面での違いがあります．

ALT キーを押した場合，コンピュータはそれを判別はしますが，文字のコードは入りません．一方，XFER の場合は，それ自体にコードが付いているので，そのままでは ALT キーのように他のキーとのコンビで使用することはできません．もし，XFER キーを ALT キーとまったく同じように使うためには，複雑で手のこんだ処理が必要です．

▶画面

画面については，IBM PC と PC-9801 との間には大きな違いがあります．

IBM PC では，前景カラーと背景カラーを属性として備えています．つまり，実際の表示色はこれら二つの色の組合せによって決まります．これによってグラフィックの色指定の多様性をもっています．一方，PC-9801 の場合は，前景カラーしかありません．このことは，IBM PC ソフトを PC-9801 ソフトとして移植するときに大きな障害となります．

逆に PC-9801 では，IBM PC と違って同時にテキストとグラフィックを表示することができます．

両機のスクリーン・バッファの違いについてはあとで詳しく説明します．

▶ I/O ポート

ハードウェア上の相違から，IBM PC の I/O ポートと PC-9801 の I/O ポートとの間には大きな違いがあります．また，アプリケーション・ソフトによっては，MS-DOS だけでなく，BIOS をもバイパスし，直接ハードウェアにアクセスすることがあります．

▶プリンタ

両機のプリンタにもいくつかの点で違いがあります．とくに IBM PC 用プリンタは ASCII 文字を中心とした IBM の基本文字をサポートし，PC-9801 用プリンタではそれ以外に ANK(英・数・かな)と漢字のサポートをしています．このため，移植したソフトでオリジナルと同じようにプリント・アウトするためには手が

〔図3〕IBM PCのスクリーン・バッファ

〔図4〕IBM PCのスクリーン・バッファの定義

```
struct IBM_VRAM_CELL {
        char c;      /* The character's extended ASCII code. */
        char attr;   /* The character's attribute.           */
};

struct IBM_CELL_ATTR {
    unsigned color: 4;       /* Character's foreground color. */
                             /* Up to 16 colors.              */
    unsigned bg_color: 3;    /* Character's background color. */
                             /* Up to 8 colors.               */
    unsigned blink: 1;       /* If ON character will blink.   */
};
```

かかります.

日本語と英語の違いによる障害

▶スクリーン・バッファの違い

両機のハードウェアの違いは，じつは英語と日本語との相違にひじょうに関係しています．つまり，日本語をサポートするために，日本のメーカが，スクリーン・バッファに関してはIBM PCとまったく違う構成を考えたのです.

移植者にとっては，移植したアプリケーションで，もとのメッセージやヘルプ・スクリーン，入力，出力，などですべて英語から日本語に直すのか，一部直すのか，などによっていろいろな問題を生じます．たとえば，アプリケーション側において，ある配列に，何文字あるかによってプログラムの流れが決まります．ところで日本語の場合は，JISを使用し，各文字に2バイトのコードが付いています．IBM PCの場合は，文字コードは1バイト・ベースになっています.

“JAPAN”の例文を考えてみると五つの文字に五つのコードとなっていますが，“日本”の例文の場合は，2文字に4バイトのコードが必要です．入力画面フィールドを使ったアプリケーションに関しては大きな問題の一つです.

〔図5〕IBM PCでの文字表示

“A”というANKの文字(0x41)と“A”というJISの文字(0x2341)を例に考えてみましょう.

▶ IBM PCのスクリーン・バッファ

IBM PCのスクリーン・バッファは，図3のように構成されています．つまり，スクリーン・バッファは，1文字ごとに文字コードと属性コードの二つが組み合わさって(この組合せを**セル**という)並んでいると考えられるのです．つまり，一つの文字に2バイトが必要で，最初のバイトが文字コード，後ろのバイトがその文字の属性コードです．属性の中には，前景カラー(4ビット)，背景カラー(3ビット)，そしてブリンク(1ビット)の三つがあります.

これらを，Cを使って定義すると図4のようになります．また，実際のメモリの中では図5のようになっています.

▶ PC-9801のスクリーン・バッファ

逆にPC-9801の場合はつぎの方式をとっています．文字のプレーンと属性のプレーンが別々になっており，各プレーンが2バイト・ユニットをベースにしています(図6)．それにより，PC-9801では，ANK文字でもJIS文字でも使用することができます(図7).

〔図6〕PC-9801のスクリーン・バッファ

〔図7〕PC-9801 での文字表示

〔図8〕PC-9801 のスクリーン・バッファの定義

```
typedef struct {
        unsigned disp_enable: 1;  /* If ON character will be shown. */
        unsigned blink: 1;        /* If ON character will blink.    */
        unsigned reverse: 1;      /* If ON character will be dis-   */
                                  /*    played in reverse.          */
        unsigned underline: 1;    /* If ON character will be dis-   */
                                  /*    played with an underline.   */
        unsigned semi_graphic: 1; /* If ON character will be dis-   */
                                  /*    played with a vertical line */
                                  /*    on it.                      */
        unsigned color: 3;        /* Character's foreground color.  */
} NEC_C_ATTR;
```

また，実際は，**図6**に示すように，各属性プレーンで，各文字ごとの属性は2バイトのユニットになっていますが，下の1バイトしか使っていませんので，**図8**のように定義することができます．

▶移植上の問題点と解決法

もし，アプリケーションが単純な表示関数を使用するだけなら，両機で同じように表示されますが，アプリケーションの多くは，画面バッファに直接アクセスするので，移植者は，文字表示を正しくするための方法を探さなくてはなりません．

さらに，ウィンドウ格納を行うアプリケーションの場合は，アプリケーションが画面ウィンドウにある文書を，バッファに一時的に保存します．そのために，アプリケーション・プログラムが，メモリのアロケートをしなくてはなりません(**図9**)．

アプリケーションは，表示データをスクリーン・バッファ上で直接変更するか，または，イメージの格納バッファ上で変更してから，そのアップデートされたウィンドウをもう一度表示します．いずれにしても，あるセルから隣のセルに行くために，ポインタを2バイトずつ先に進めなくてはなりません．

IBM PC のアプリケーションの場合，**図10**のような定義を使うことがあります．この定義を使ったアプリケーション・ソフトのプログラム中には，

pvcell++ とか (pvcell+1)->attr

のような命令がたくさん出てきます．そこで，こういったアプリケーション・ソフトを移植するときには，プログラム中の各ポインタを修正していくよりも，アプリケーションの基本ロジックを拡張するという考え，つまり，**図10**の定義を**図11**のように変えます．この方法を用いると，あとのデバッグの作業がずっとらくになります．

しかし，この方法にもまだ問題があります．IBM PC の場合は，アプリケーション用ウィンドウのイメージと，実際の画面バッファのとは同じ構成なので，イメージをそのままバッファに入れてしまうと，文字と属性のコードが一緒に表示されてしまいます．一方，PC-9801 の場合は，文字と属性は別々になっています．つまり，移植者は，イメージを読み込むときに，文字と属性とをそれぞれ別々に整理しなくてはなりません．

▶ gettext と puttext の問題点

こういったテキスト・ウィンドウを画面から直接読み込むアプリケーションはたくさんあります．たとえば，米ボーランド社の Turbo C を用いたアプリケーションの場合には，gettext と puttext の関数をよく使用します．ここで注意しなくてはならないことがあり

〔図9〕IBM PC ソフトを PC-9801 で表示するには

〔図10〕IBM PC 用セルの定義

```
struct IBM_VRAM_CELL {
        char c;
        char attr;
} *pvcell;
```

〔図11〕PC-9801 用セルの定義

```
struct NEC_VRAM_CELL {
        unsigned c;
        unsigned attr;
} *pvcell;
```

ます．つまり，Turbo C コンパイラ自体も移植されたものであるということです．アプリケーションの移植者が PC-9801 用に合わせるときには，双方のコンパイラ間にも相違があることを忘れてはなりません．

具体的に述べましょう．IBM PC の場合，puttext と gettext の命令は，

文字，属性，文字，属性，文字，属性，文字，…

のようにして格納バッファに格納します．一方，PC-9801 の場合は，1 行ごとに，

文字，文字，文字，…，属性，属性，属性，…

といったように，それぞれ文字と属性が行のなかでまとめられて行われます．この差が，アプリケーションの移植者に負担を重くしているといえます．この点で，Turbo C の移植は，アプリケーションの移植に対しての配慮が足らなかったといえると思います．

この部分をアプリケーション・プログラム上で直そうとするととてもたいへんなので，筆者は，コンパイラのこの puttext と gettext の部分だけを自分で作り直しました．このほうがずっと時間の節約になります．

半角文字と全角文字の問題

▶ 1 バイト・コードと 2 バイト・コード

他にもまだまだ注意しなくてはならないことがいっぱいあります．

IBM PC の場合は，スクリーン・バッファの各セルが独立しており，セル同士の影響はありません．たとえば，直接セルに違う文字を書いた場合，画面にその文字が表示され，周囲の文字はまったく影響されません．

PC-9801 の場合にはそういうわけにはいきません．つまり，日本語をサポートするために，文字用プレー

〔表 1〕IBM PC または PC-9801 で表示できる文字セット例

(a) JIS 全角コードの例

CODE (HEX)	2341	2342	2343	2344	4D2D
全角 (JIS)	Ａ	Ｂ	Ｃ	Ｄ	有

(b) JIS 半角コードの例

CODE (HEX)	2941	2942	2943	2944	2A61
半角 (JIS)	A	B	C	D	ｴ

(c) ANK コードの例

CODE (HEX)	41	42	43	44	C9	BB
ANK	A	B	C	D	ﾉ	ｻ

(d) ASCII コードの例

CODE (HEX)	41	42	43	44	C9	BB
ASCII	A	B	C	D	╔	╗

〔図 12〕文字列コード例

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	`	p	Ç	É	á	░	└	╨	∝	≡
1	☺	◄	!	1	A	Q	a	q	ü	æ	í	▒	┴	╤	β	±
2	☻	↕	"	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
3	♥	‼	#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5	♣	§	%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6	♠	▬	&	6	F	V	f	v	å	û	ª	╢	╞	╓	µ	÷
7	•	↨	'	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿	╕	╚	╪	Φ	°
9	○	↓	)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	θ	•
A	◙	→	*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
B	♂	←	+	;	K	[	k	{	ï	¢	½	╗	╦	█	δ	√
C	♀	∟	,	<	L	\	l	¦	î	£	¼	╝	╠	▄	∞	n
D	♪	↔	—	=	M	]	m	}	ì	¥	¡	╜	═	▌	φ	2
E	♫	▲	.	>	N	^	n	~	Ä	₧	«	╛	╬	▐	∈	■
F	☼	▼	/	?	O	_	o	△	Å	ƒ	»	┐	╧	▀	∩	BLANK FF

(a) IBM PC のキャラクタ・コード

(b) PC-9801 のキャラクタ・コード

(c) 漢字 JIS コード

ンの中で，１文字２バイトの間に半角の位置があるからです．もう少し具体的にみてみましょう．

漢字は正方形になっているために，表示するときにASCII文字を２個合わせたものと同じ面積を必要とします．漢字１文字は普通，全角と呼ばれ，ASCIIやANKのような漢字の半分のスペースですむ文字は半角とよばれます(**表１**)．

▶半角文字とは何か

PC-9801用ソフトの日本語部分の多くはJISコードの全角文字を使用しています．しかし，JISコードのなかには半角文字もセットで用意されています．これは，２バイト・コードでありながら半角(縦長の長方形)で表示されます．つまり，**表１**を見てもわかるように，“A”という文字のサイズは，半角セット，ANK，ASCII，の３種類とも同じですが，このうち半角セットの“A”とANKの“A”とは，違うフォントということです．移植者は，半角文字についていうときは，それが半角セットの文字なのか，それと一般的に半角に表示される文字なのか，よく注意しなくてはなりません(**図12**)．

このこと以外にも，最近ではパソコン用日本語ワープロ・ソフトのなかには，半角の漢字フォントをもつものまであります(P1EXE，一太郎ver 4など)．

▶半角文字と全角文字の混在

PC-9801の場合は，ANK文字を画面に直接表示させたいときには，１バイト・コードを２バイト・コードに拡張しなくてはなりません．たとえば，Cプログラムで“A”というANK文字を直接表示するときは，

A(0x41) → A(0x0041)

と，２バイトに拡張します．逆にいうと，“A”という文字は，画面上に表示するときに，その文字のセルの内容が4100のようになっているということです．

したがって，画面上の各半角の位置に関しては，スクリーン・バッファに２バイト・コードが対応していることになります．この論理にそっていえば，全角文字には，スクリーン・バッファに４バイト・コードが必要となります．実際そのとおりで，全角文字を画面に直接表示したい場合は，文字コードから表示用コードへの変換ルーチンが必要になってきます．JIS文字の変換ルーチンは**図13**のようになります．

〔図13〕JISコード → 表示用コード変換ルーチン

```
unsigned long
jis2dsp( unsigned int jis_code )
{
            _DX = jis_code;

    asm     MOV     AL, DH
    asm     SUB     AL, 20H
    asm     MOV     AH, DL

    asm     XCHG    DL, DH
    asm     ADD     DL, 60H
}
```

このように，PC-9801では(日本語サポートのマシンであれば)半角文字と全角文字が混在してしまうので，あるテキスト画面の上に別のウィンドウ画面をかぶせたいときに問題が生じます．このとき，下になる画面の情報は画面バッファから別のメモリ領域に退避させなくてはなりませんが，問題は，ウィンドウ画面の境

〔図14〕漢字と英字の混在の問題 ―― ウィンドウの例

日本人日本人日本人日本人日本
Japanese Japanese Japanese
この位置にウィンドウを開く
かぶせたいウィンドウ
“J”が隠れる
〈単純に行うとこうなります〉
日apanese Japanese Japanese本
“本”がずれる

(a) 漢字文字列の中に欧文のウィンドウを挿入する

日 本 人 半角スペース
26 7C A6 76 2B 5C AB 5C 1F AD 9F 4D 2A 00
“日”の半角の位置に“Ja”が入るとする
J a
4A 00 61 00
4A00と変化しても267Cが変化しないと，(ハードウェア機構により)“日”の文字が残る
26 7C 4A 00 61 00 2B 5C AB 5C 1F AD 9F 4D
日 a 本 人
〈コード〉
日本人_
Ja
日a本人_
半角ぶんずれる
〈表示〉

(b) “日”のまん中に“Ja”が入るメカニズム

〔図15〕外字フォントの設計例とコード

〔図16〕外字フォントを BIOS をとおして外字バッファに
登録する

```
/*-----------------------------------------
SET_GAIJ: 16×16ドットの外字フォントを、
          BIOSをとおして外字バッファに
          登録する。

          *font:  フォント格納のアドレス
           code:  外字コード（機種によって
                  7621H ～ 765FH  または
                  7621H ～ 767EH  または
                  7721H ～ 777EH  ）
-----------------------------------------*/
void
set_gaij( unsigned *font, unsigned code )
{
    asm   MOV DX, code

#if defined(__TINY__) || defined(__SMALL__)
      || defined(__MEDIUM__)
    asm   MOV CX, font
    asm   MOV BX, DS
#else
    asm   LES CX, font
    asm   MOV BX, ES
#endif
    asm   MOV AH, 1AH
    asm   INT 18H
}
```

界線上で起こります．つまり，境界線が下の画面の文字上にどうしてもくることがありうるのです(**図14**)．このとき，文字のコードが切れてしまうので，表示がおかしくなってしまいます．

▶ IBM PC の基本文字セットの問題点

先に示した**図12**と**表1**を見てください．PC-9801 と IBM PC とは，ASCII 文字セットに関して，基本的に同じものを使用しているのですが，文字セット表の上半分の 127 文字(80H～FEH)はまったく違っています．また，下の 00H から 1FH までも違っています．ここで，IBM PC のアプリケーションが，PC-9801 で表示できない文字セットを用いている場合，なんらかの代替表示法を考えなくてはなりません．

たとえば，IBM PC にある ASCII の“╔”と“╗”のような文字は，IBM PC 用アプリケーションでとても頻繁に使用されるものなのですが，PC-9801 の使用する文字コードにはありません．

一つのアプローチは，これとまったく同じでなくとも，できるだけ似ている文字を用いるという方法です．たとえば先の例では，“╔”の代わりに“「”を用いるわけです．ただし，ここで注意しなくてはならないのは，たとえば“╔”とペアで用いられる“《”の代わりに“く”を用いようとしても，すでに“く”が別の意味で使われていることがあったりする，ということです．

▶半角の解決策

賢い移植者なら，ここで PC-9801 の半角コードを用いることでしょう．この半角コードは，JIS 全角文字と同じ 2 バイト・コードですが，画面上では ANK 文字と同様に扱うことができます．たとえば，《の文字は，その半角 2 バイト・コードにもあります(JIS コード 0x2B78)．ただ，この手法もオールマイティではありません．たとえば，IBM PC にある § の文字は，ANK にも半角文字セットにも見つけることはできません．移植者は，このようなとき以下のような方法を考えるでしょう．

(1) グラフィック・エミュレータを用いる

テキスト画面をやめて，PC-9801 のグラフィック画面で IBM PC 用のフォントを使用し，IBM PC のテキスト画面をエミュレートします．ただし，このアプローチでは，アプリケーションのパフォーマンスが低下すると考えてよいでしょう．

(2) 外字を使用する

IBM PC にしかない文字を PC-9801 用の外字として用意する方法です．つまり，PC-9801 にある外字バッファも用いてテキスト画面に表示します．

外字バッファは，ユーザ定義フォントのために，JIS コードの中にいくつか用意されており，外字コードにいろいろなフォントを入れることができます．本来の目的は，日本語の文章のなかで出てくる特別の記号や JIS の標準文字にない文字などを登録するためにあり，フォント・サイズは 16×16 ドットなので，基本的に全角文字として表示されます．

フォントの左半分だけを使って半角のようにも表示できますが，それは，外字のつぎに ANK を出したと

〔図17〕外字フォントを直接ハードウェアにアクセスして外字バッファに登録する

```
/*--------------------------------------------------
SET_GAIJ: 16×16ドットの外字フォントを、ハードに
          直接アクセスして、外字バッファに登録する。

          *font:  フォント格納のアドレス
           code:  外字コード（機種によって
                  7621H ～ 765FH  または
                  7621H ～ 767EH  または
                  7721H ～ 777EH  ）
--------------------------------------------------/*
void
set_gaij( char far *font_addr, unsigned code )
{
  asm  MOV AL, 0BH      /*;Dot map select command      */
  asm  OUT 68H, AL      /*;Set CG code                 */

     SET_JIS_CODE:

  asm  MOV AX, WORD PTR code
  asm  OUT 0A1H, AL     /*;Set JIS low code in CG      */
  asm  MOV AL, AH
  asm  SUB AL, 20H
  asm  OUT 0A3H, AL     /*;Set JIS high code(-20H)     */

     SET_INITIAL_VALUES:

  asm  LES SI, font_addr
  asm  MOV DL, 0  /*;Set font's line counter to 1st line*/

  asm  MOV CX. 16 /*;Set read loop counter                  */

     NEXT_FONT_LINE:

  asm  MOV AL, DL         /*;Get next font line no.       */
  asm  MOV BX, ES:[SI]    /*;Get font line from buffer    */

  asm  OR AL, LEFT_MARK   /*;Set L/R bit to ON (left)     */
  asm  OUT 0A5H, AL       /*;Ask to get line's left part  */
  asm  MOV AL, BH         /*;Get left font of line        */
  asm  OUT 0A9H, AL       /*;Set left font of line        */
  asm  MOV AL, DL         /*;Get again font line no.      */
  asm  OUT 0A5H, AL       /*;Ask to get line's right part*/
  asm  MOV AL, BL         /*;Get right font of line       */
  asm  OUT 0A9H, AL       /*;Set right font of line       */

  asm  INC SI                /*;Advance to next font line*/
  asm  INC SI                /*;(i.e. next 16 bit word)  */
  asm  INC DL                /*;Advance font line counter*/
  asm  LOOP NEXT_FONT_LINE   /*;Go to next font line     */

     OUT:

  asm  MOV AL, 0AH           /*;Code map select command  */
  asm  OUT 68H, AL           /*;Set CG mode              */
}
```

きしか有効ではありません．つまり，§A§A§A はできますが，このままでは§§§のようには続けられません．このときは，§§§のように表示されます．

半角の場合は，くくくの ANK のように，《《《は連続して表示できます．しかし，ここにも悩みの種があります．

一つの解決法は，つぎのように考えられます．文字表示の前に，前回表示した文字を確認しても，もし外字でなければ，そのまま表示することができます．外字の場合は，違う外字コードを使って，前回だした文字フォントと，これから出したい文字のフォントを合成して，とりあえず違う文字に登録します．その外字を表示する，たとえば，§のつぎにもう一度§を出したい場合は，§§を一つのフォントとして外字登録してから，その外字を表示します．

外字の登録個数は，機種によって 63，94，188 となりますが，これらの外字をすべて使いきってしまった場合の注意があります．それは，外字登録されるときに，画面全体がリフレッシュされますから，以前に登録したコードと同一のコードを登録した場合，前回の外字は消失し，新しいものに入れ替わってしまうということです．外字フォントの登録法を**図15～17** に示しておきます．

まとめ

以上，ソフトの移植者がかかえる問題点のうち，いくつかを取り上げ，その解決策をどのようにしてみつけるかを説明しました．筆者の考えでは，ポーティングは，テクニックというよりもアートに近いもののように思えます．きまったルールがなく，いろいろなケースによって，その処理の方法が異なってきます．

ある意味で移植者は，異文化の人たちの間にたつ通訳者でもあります．通訳するときは，たんなる翻訳だけではなく，それぞれの文化の違いを考慮しなくてはなりません．

イーガル・ブルーメンライヒ
キュー・エックス・エンタープライズ㈲

筆者紹介

イスラエル生まれ．大学および大学院で生命工学を専攻しながら，博士課程で生理学，コンピュータを学ぶ．8年前に来日．日本人女性と結婚し，現在都内に在住．5年前に日本においてソフトハウスを設立．大型からパソコンまでのソフト開発を行う．現在，PC-9801用ソフトの開発，海外ソフトの移植，CD-ROM用ソフトの開発などとともに，コンサルタント業務も手掛ける．

●特別企画

トランスピュータ・ボードの製作 [下]

Occam によるマルチプロセシングを可能とする

那須川 徳博

実際に製作した2種のトランスピュータ・ボードのハードウェアについて解説する．サブ・トランスピュータには2個のT414と，それぞれに128Kバイトの SRAMを搭載した．TDS2を動作させるためのマスタ・トランスピュータは，T414と1MバイトのDRAM で構成している．ソフトウェアは，メモリ回路のチェックを行うためのプログラムと，Occamの動作を確認するためのマンデルブロ曲線の描画プログラムについて解説する．（編集部）

前号では，トランスピュータの概要や信号のタイミングなどについて説明してきました．今回はその後編として，実際の回路設計と開発環境や Occam のプログラミングについて説明していきます．

トランスピュータでは，メモリ・アクセス信号のタイミングをメモリの種類に応じて変えることができるため，回路設計が容易で外付け回路のゲート数を最小にすることができます．設計した回路では，ダイナミック RAM を使用してもスタティック RAM の場合とあまり変わらないハードウェア量で構成することができました．

マイクロプロセッサにダイナミック RAM を接続する場合，設計方法によってはメモリ・アクセスのスループットに大きな差が生じることがあります．トランスピュータではリフレッシュ回路を内蔵したことで，リフレッシュおよびリフレッシュとメモリ・アクセスの競合問題を考慮する必要なく，誰が回路設計を行っても高いパフォーマンスを得ることができます．設計上の注意点といえば，メモリ・アクセス・タイムと動作クロック周波数が高いのでゲートの遅延時間に注意しなければならないことくらいです．

製作したマスタ・トランスピュータ・ボードは，単独でも高性能の32ビット・プロセッサとして使用することができ，さらにサブ・トランスピュータ・ボードを多数接続することで大型汎用計算機の処理能力にも匹敵する性能を発揮できる可能性をもっています．

3 トランスピュータ・システムの回路設計

プログラム開発を容易にするためには，既存の開発環境があると便利ですが，トランスピュータではインモスによって開発された TDS2(Transputer Development System)と呼ばれる開発システムがあります．

TDS2 は，Occam 言語コンパイラを含み，エディタ，デバッガなどトランスピュータのプログラム開発に必要なユーティリティを備えた独特の開発環境です．

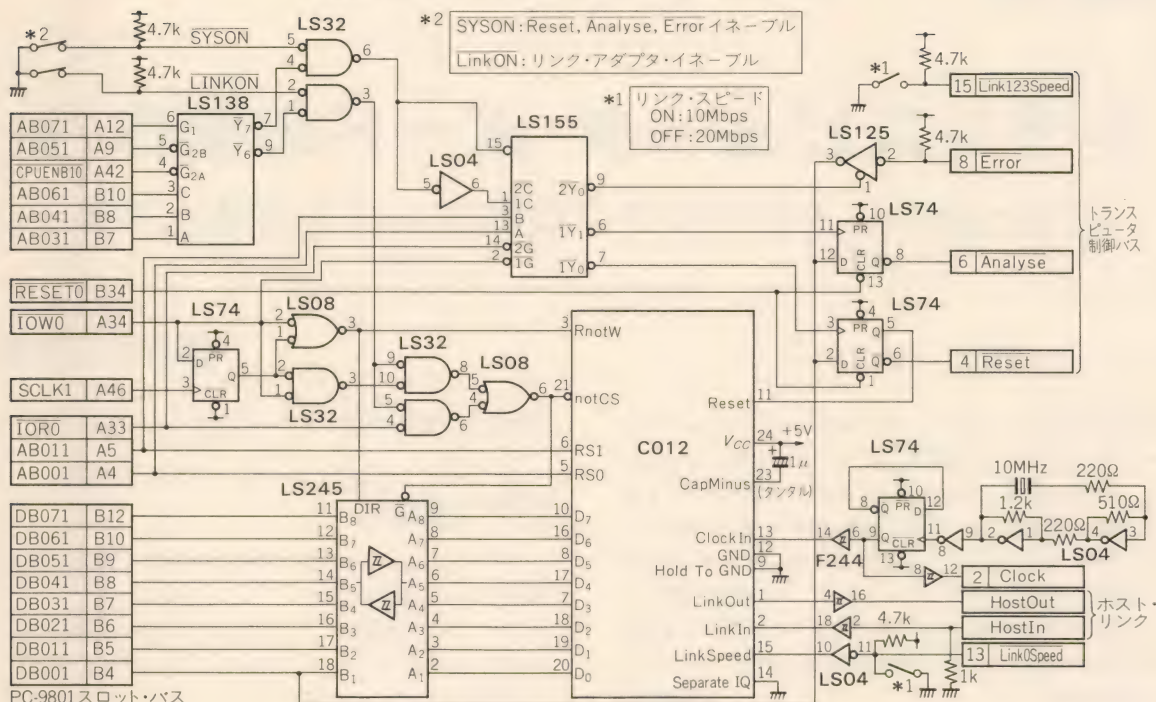
トランスピュータと Occam は切っても切れない関係にあり，そのためにも TDS2 を使用することが不可欠です．インモスでは，PC-9801 で使用できるトランスピュータ・ボードと TDS2 を発売していますが，今回製作したボードも PC-9801 版の TDS2 を実行できるようなハードウェア構成としました．

3.1 ハードウェア構成

トランスピュータで動作するプログラムは，TDS2 によってシリアル・リンクからロードされるため，トランスピュータのメモリは，RAM だけで構成することができます．

トランスピュータ・システムの中心となるマスタ・トランスピュータには，20 MHz の T414 を使用し，ダイナミック RAM により1Mバイトのメモリ容量とします．このマスタ・トランスピュータは C012 リンク・アダプタにより PC-9801 と接続され，プログラムのコンパイル，サブ・トランスピュータの制御やプログラムのロードなどを行います．

サブ・トランスピュータには，マスタ・トランスピュータと同じく 20 MHz の T414 を使用し，スタティック RAM により 128K バイトのメモリ容量とします．サブ・トランスピュータは，マスタ・トランスピュータの制御下に置かれ，マスタ・トランスピュータ

〔図23〕PC-9801とトランスピュータのインターフェース

から送られてきたプログラムやデータを受け取り実行します．

これらマスタとサブのトランスピュータのハードウェアは，それぞれPC-9801の拡張スロットに差し込む拡張ボード上に組み込んでおり，マスタ・トランスピュータ・ボードにはリンク・アダプタなどPC-9801とのインターフェースが含まれています．

またサブ・トランスピュータは，1枚のボード上に2組のトランスピュータを搭載しています．このサブ・トランスピュータ・ボードは，PC-9801の拡張スロットからは電源しか使用していませんので，同じボードを複数枚使用してリンクで接続することにより，さらに処理能力を向上させることができます．

3.2 トランスピュータとホストのインターフェース

トランスピュータ・システムの多くは，トランスピュータに直接コンソールやハード・ディスクを接続するのではなく，接続するパーソナル・コンピュータの資源を利用し，パーソナル・コンピュータのマイクロプロセッサがその資源をアクセスしてトランスピュータに渡します．つまり，パーソナル・コンピュータのマイクロプロセッサが，トランスピュータのI/Oプロセッサになります．

TDS2は，トランスピュータ側で動作するTDS2プログラム本体と，パーソナル・コンピュータ側で動作し，トランスピュータからのコマンドを解析するサーバ・プログラムに分かれ，コンソール，ハード・ディスクなど機種により異なるI/O部分は，サーバ・プログラム側によって吸収しています．

このサーバ・プログラムは，MS-DOS上で動作しており，トランスピュータからのコマンドを解析し，MS-DOSのシステム・コールに変換します．

TDS2のいちばん新しいバージョンであるD700Dバージョンでは，PC-9801用にNECSERVE.EXEというサーバ・プログラムがありますが，試作したハードウェアではこのサーバ・プログラムで使用しているリンク・アダプタやReset, Analyse信号出力のI/Oアドレスに合わせることによって，TDS2が動作するようにしています．

PC-9801版のTDS2のI/Oアドレスが使用できない場合や，PC-9801以外の機種でTDS2を使用してトランスピュータを動作させるためには，サーバ・プログラムのソース内のリンク・アダプタやReset，Analyse制御のI/Oアドレスを変更する必要があります．

D700Dバージョンでは，サーバ・プログラムのI/O制御部分であるCSERVER.TSR(C言語のソース)かIOASM.TSR(アセンブラ言語のソース)をコンパイル

〔表 9〕 I/O アドレス・マップ

アドレス	リード/ライト	項　目
D0H	R	リンク・データ入力
D2H	W	リンク・データ出力
D4H	R	リンク・ステータス・リード
D6H	R	リンク・ステータス・ライト
D8H	R	リンク・エラー・ステータス入力
D8H	W	Reset 出力
DAH	W	Analyse 出力

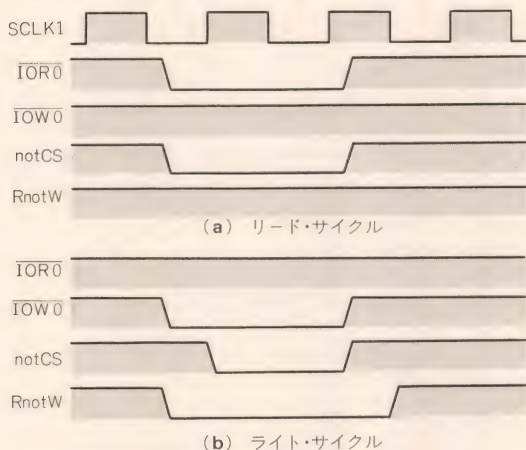
〔図24〕 リンク・アダプタのアクセス・タイミング

またはアセンブルし直すことにより，他の機種に移植することが可能です．

図23 は，PC-9801 とリンク・アダプタとのインターフェース回路です．PC-9801 とトランスピュータ間を接続するリンク・アダプタ，Reset と Analyse 出力，Error フラグ入力は I/O 空間に割り当てられています．アドレスの割当ては**表 9** のようになります．

このなかで，D0H～D6H が C012 リンク・アダプタに対するデータやステータスのアクセスで，D8H～DAH はマスタやサブのトランスピュータの Reset と Analyse の出力，Error ステータスの読出しに使用されます．

図24 は，リンク・アダプタ・インターフェースのアクセス・タイミングです．C012 のチップ・セレクト notCS は，PC-9801 の $\overline{IOW0}$ と $\overline{IOR0}$ により作り出します．リード・サイクルでは $\overline{IOR0}$ をそのまま加え，ライト・サイクルでは $\overline{IOW0}$ 信号を PC-9801 のシステム・クロックである SCLK1 の立上りエッジによって遅らせて加えています．これは，ライト・サイクルで RnotW 信号は，notCS 信号がアクティブになる前にアクティブになっていなければならない (RnotW のセットアップ時間：$T_{RLCL}>5$ ns) ためです．

▶クロック

トランスピュータ・ファミリのデバイスでは，5 MHz のクロックを加える必要がありますが，そのために 10 MHz のクロックを発生させ，D フリップフロップにより 2 分周して 5 MHz のクロックを得ています．5 MHz のクロックは，立上り，立下りエッジの時間が問題になり，立上り時間が 10 ns，立下り時間は 8 ns 以下であることが要求されます．そこで，シュミット・トリガ・タイプのバッファを使用して，リンク・アダプタやトランスピュータへ与えています．

▶ Reset，Analyse，Error

Reset や Analyse 信号は，トランスピュータのリセットやエラー解析に用いられますが，D フリップフロップを用いて出力が保持されるようにします．

またこれらの信号は，電源 ON 時には PC-9801 のリセット信号により出力が“H”で非アクティブの状態になるようにします．これらの信号の出力は，Reset が D8H 番地，Analyse が DAH 番地へのデータ書込みにより行われ，ビット 0 に“1”を書き込むことによりアクティブ，“0”を書き込むことで非アクティブになります．

また Error フラグは D8H 番地からのデータ読出しにより読み出され，ビット 0 が“0”なら Error フラグが立っていることを示します．

▶シリアル・リンク

シリアル・リンクの入出力はバッファを使用しています．シリアル・リンクの入出力信号は，立上り/立下り時間ともに 20 ns 以下であることが要求されるため，F タイプの TTL を使用します．

バッファの入力は，入力オープンでは“H”の入力状態になりますので，1 kΩ のプルダウン抵抗を GND 間に接続し，リンク入力がオープンでも“L”レベルになるようにします．

▶その他の端子

LinkSpeed：シリアル・リンクのスピード選択のために用いられ，この入力はほかのトランスピュータの Link0Speed 入力といっしょに接続し，スピードの切替えに使用します．

SeparateIQ：この端子は GND に接続します．

CapMinus：この端子は，内部クロック信号を発生する PLL 回路のためのデカップリング・コンデンサ接続端子で，1 μF 程度の周波数特性の良いコンデンサ (タンタル・コンデンサなど) を V_{CC} とこの CapMinus 端子間に接続します．

3.3 サブ・トランスピュータの回路構成

まず，回路構成が簡単なスタティック RAM によるサブ・トランスピュータ回路の構成について説明していきます．サブ・トランスピュータでは，

(1) 回路構成をコンパクトにできる

〔図25〕サブ・トランスピュータの回路

(2) マスタ・トランスピュータのようにコンパイラを動作させる必要がなく，メモリが小容量でもよい

(3) ダイナミック RAM に比べて高速に動作させることができる

などの理由により，スタティック RAM を使用することとしました．**図25** は，そのサブ・トランスピュータの回路で**写真 1** はそのボードの外観です．

使用したメモリは，メモリ容量を 128 K バイトとするために 32 K×8 バイトのスタティック RAM HM-62256-10(アクセス・タイム 100 ns)を 4 個使用しています．

トランスピュータの制御は，システム・クロック，Error, Reset, Analyse と Link0Speed, Link123Speed の各信号により制御可能で，データの通信はシリアル・リンクにより必要なリンクだけを接続します．

▶ Reset，Analyse，Error のインターフェース

Reset，Analyse，Link0Speed，Link123speed およびクロックは，PC-9801 インターフェースから出力される信号をそのまま接続するだけです．これらの入出力信号は負論理で入出力します．

Error 出力は複数のトランスピュータからの出力を接続しますので，ワイヤード OR 接続可能なオープン・コレクタ型のゲート 7406 を使用しています．

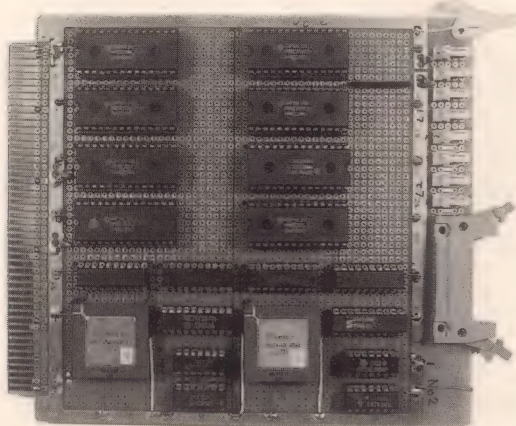
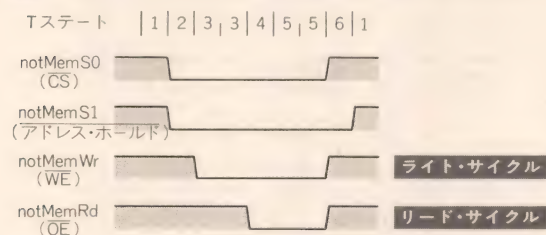
▶その他の端子

MemReq は DMA によるメモリ・アクセス要求，EventReq はイベント要求，MemWait はメモリ・アクセスのウェイト・サイクル挿入ですが，今回は使用しませんので GND に接続します．また，BootFromRom はプログラムを ROM からブートするかシリアル・リンクからブートするかを選択しますが，シリアル・リンクからプログラムをブートしますので GND に接続します．

LinkSpecial はリンク・スピードの選択ですが，ここではプルアップ抵抗で電源に接続し，リンク・スピードを 10 Mbps または 20 Mbps の選択としています．

●外部メモリ・インターフェースの設計

トランスピュータでは，アドレス出力とデータの入出力が時分割で行われますが，まず D ラッチを使用してデータの入出力までアドレスを保持する必要があります．トランスピュータでは，メモリのアクセスのために notMemS0～4 までの汎用ストローブ信号があり，notMemS1 はサイクルの最後 T_6 まで出力されますので，この信号でアドレスをラッチします．

〔写真1〕サブ・トランスピュータ・ボードの外観

〔図26〕サブ・トランスピュータのアクセス・タイミング

また，スタティック RAM のチップ・セレクトは，AD31 と notMemS0 の AND をとり簡易デコードしています．

HM62256 の読出しには $\overline{\mathrm{OE}}$，また書込みは $\overline{\mathrm{WE}}$ を使用しますが，この端子に加える信号としてトランスピュータでは，読出しのために notMemRd，書込みのために notMemWrB0～3 があります．

読出し信号は，すべてのチップに同時に加えていますが，トランスピュータ内部で必要なバンクのデータだけが内部で読み込まれます．

書込みは，バイト単位での書込みも行われるため，notMemWrB0 が D0～7，notMemWrB1 が D8～15，notMemWrB2 が D16～23，notMemWrB3 が D24～31 にそれぞれ対応するよう接続します．

外部メモリのアクセス・タイミング設定は，MemConfig 端子をデータ・バスに接続し，リセット時にタイミング発生を設定します．スタティック RAM のためのタイミングとして，3プロセッサ・サイクルと4プロセッサ・サイクルの2種類が，内部コンフィギュレーションとして用意されていますが，ここでは4プロセッサ・サイクルに設定しています．そのために，MemConfig 端子は MemAD9 に接続します(前号 p.267 参照)．

〔写真2〕マスタ・トランスピュータ・ボードの外観

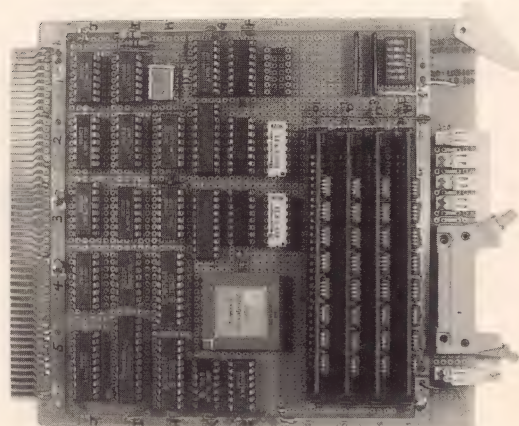



図26 は外部メモリ・アクセスのタイムチャートです．動作クロックが 20 MHz のトランスピュータでは，1サイクルの周期が 50 ns ですので，一度のメモリ・アクセス時間は4プロセッサ・サイクルでは 50×4＝200 ns となります．また T ステートはクロック周期の 1/2 ですから 25 ns となります．

HM62256-10 のアドレス入力からのアドレス・アクセス時間(t_{AA})は 100 ns, $\overline{\mathrm{OE}}$ からの出力イネーブル・アクセス時間(t_{OE})は50nsです．データの読込みは，T_5 ステートの終わりで行われますので，アドレスが出力されてから 175 ns 後，$\overline{\mathrm{OE}}$ がアクティブになってから 75 ns 後になります．

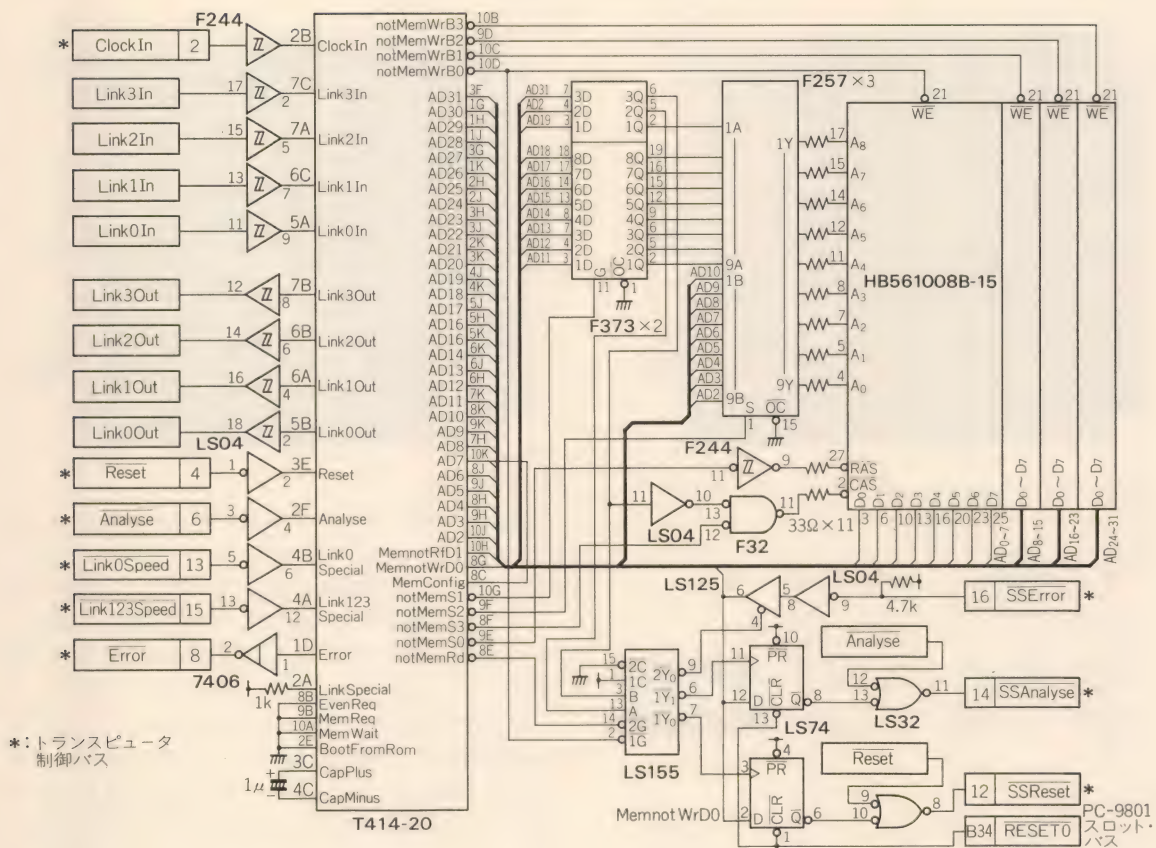
3.4　マスタ・トランスピュータの回路構成

図27 はマスタ・トランスピュータの回路で，**写真2** は PC-9801 とのインターフェースも含めたマスタ・トランスピュータの外観です．

マスタ・トランスピュータはダイナミック RAM を使用して回路を構成しています．使用したダイナミック RAM は，256 K×8 ビットのダイナミック RAM モジュール HB561008B-150(アクセス・タイム 150 ns)を4個使用し，1M バイトのメモリ容量としています．

ダイナミック RAM を使用したメモリの回路設計では，$\overline{\mathrm{RAS}}$ や $\overline{\mathrm{CAS}}$ 信号の作成，アドレスの切替え，メモリのリフレッシュなどを考えて設計しなければなりませんが，トランスピュータでは，すべてこれらの信号を出力してくれるので，回路が非常に簡単になります．

ダイナミック RAM のアクセスは，ロウ・アドレス出力→ $\overline{\mathrm{RAS}}$ 出力→カラム・アドレス出力→ $\overline{\mathrm{CAS}}$ 出力→データの読込み(あるいは書込み)の順番で行われます．

〔図27〕マスタ・トランスピュータの回路

まずアドレスですが，最初にセレクタ 74F257 により MemAD2～10 をダイナミック RAM に与えます．このアドレスはロウ・アドレスですから $\overline{\text{RAS}}$ 信号がアクティブになるまでの間だけ保持すればよく，D ラッチによりメモリ・アクセス・サイクルの終わりまで保持する必要がありません．

つぎに notMemS0 を $\overline{\text{RAS}}$ としてダイナミック RAM に与え，それと同時にカラム・アドレスとして notMemAD11～19 を D ラッチにより保持します．

そして，notMemS2 の立下りによりアドレスをロウ・アドレスからカラム・アドレスに切り替えてダイナミック RAM に与え，最後に $\overline{\text{CAS}}$ を notMemS3 により与えます．そして $\overline{\text{WE}}$ がアクティブかどうかによりデータの読出しまたは書込みが行われます．

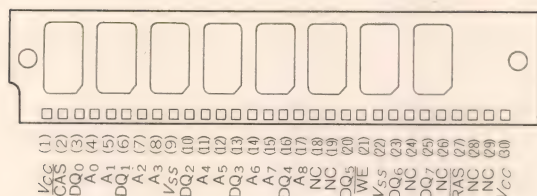
ダイナミック RAM はリフレッシュも必要ですが，トランスピュータでは ClockIn 入力の 72 クロック周期で行われますので，200 ns×72＝14.4 μs ごとにリフレッシュが行われます．リフレッシュは，リフレッシュ・アドレスを与えて $\overline{\text{RAS}}$ 信号だけをアクティブにする RAS オンリ・リフレッシュ方式により行います．

リフレッシュ時には，notMemS0～4 のタイミングは，通常のメモリ・アクセスと同じく出力されます．そのため，通常のメモリ・アクセスと区別する必要がありますが，リフレッシュ時には AD31 の出力が“L”になるため，notMemS3 と AD31 の AND をとり $\overline{\text{CAS}}$ とし，リフレッシュ時に $\overline{\text{CAS}}$ 信号が出力されないようにしています．またこの AD31 は，簡易デコードの役目も兼ねています．

リフレッシュ時には，AD2～11 にリフレッシュ・アドレスが出力されます．このアドレスは，通常のメモリ・アクセス・サイクルと同じくロウ・アドレス・ホールド時間（t_{RAH}＞15 ns）だけ保持すればよいので，D ラッチは入れていません．

アクセス・タイミングは，MemConfig を MemAD7 に接続し，6 プロセッサ・サイクルのダイナミック RAM に設定します．このタイミングでは書込み信号が早めに出力されるアーリ・ライトのため，ダイナミック RAM の入力と出力をいっしょに接続することができます．

HB561008B-150 のピン配置を**図28** に示します．ダイナミック RAM は，おもなタイミングとして**表10** のようなタイミングが要求され，6 プロセッサ・サイク

〔図28〕 HB561008B のピン配置

〔表10〕 HB561008B-150 のタイミング(単位：ns)

記号	項　　目	min	max
t_{RC}	ランダム・リード・ライト・サイクル時間	260	
t_{RCD}	RAS/CAS 遅延時間	35	
t_{RAC}	RAS からのアクセス時間		150
t_{CAC}	CAS からのアクセス時間		75
t_{RP}	RAS プリチャージ時間		100
t_{RAS}	RAS パルス幅	150	
t_{CAS}	CAS パルス幅	75	
t_{RAH}	ロウ・アドレス・ホールド時間	15	
t_{CAH}	カラム・アドレス・ホールド時間	25	
t_{WP}	ライト・コマンド・パルス幅	45	

ルのダイナミック RAM サイクルでは**図29** に示したタイミングになります.

アクセスのサイクルは 300 ns になり，データの読込み/書込みは T_5の終わりで行われますから，$\overline{\text{RAS}}$ からのアクセス・タイムは 200 ns 確保されています.

なお，アドレスや $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ などの信号はアンダシュート，オーバシュートを減らして誤動作を防ぐために 33 Ω の抵抗を直列に挿入しました.

▶ Reset, Analyse, Error

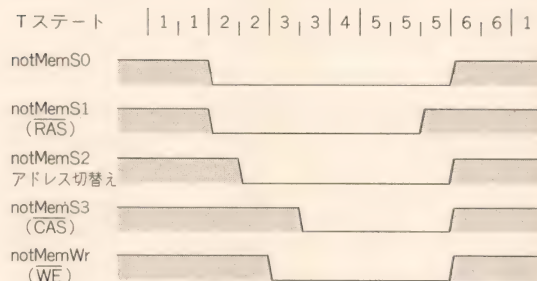
システム・クロック，$\overline{\text{Reset}}$, $\overline{\text{Analyse}}$, $\overline{\text{Link0Speed}}$, $\overline{\text{Link123Speed}}$ は，PC-9801 からの出力をそのまま接続します. $\overline{\text{Error}}$ は，サブ・トランスピュータと同じくオープン・コレクタ型のゲートを使用します.

これら制御信号に加えてマスタ・トランスピュータでは，サブ・トランスピュータを制御するための信号として $\overline{\text{SSReset}}$, $\overline{\text{SSAnalyse}}$, $\overline{\text{SSError}}$, の入出力を備えています.

ここで SS××という信号は，マスタ・トランスピュータからの命令によってサブ・トランスピュータを制御するためのもので，これらの信号は PC-9801 からの出力 Reset, Analyse と OR 接続しています.

サブ・トランスピュータは PC-9801 からのみ制御するか，またはトランスピュータからの制御も加えるかによって $\overline{\text{SSReset}}$, $\overline{\text{SSAnalyse}}$, $\overline{\text{SSError}}$ か $\overline{\text{Reset}}$, $\overline{\text{Analyse}}$, $\overline{\text{Error}}$ のいずれかを使用します.

トランスピュータを増設するためのバスは，**図30** のような簡単なバスとしましたが，各社から発売されているトランスピュータ・ボードでは互換性をとるため

〔図29〕 マスタ・トランスピュータのアクセス・タイミング

〔図30〕 トランスピュータ制御バスの信号配置

Clock	2	1 ◀	GND
$\overline{\text{Reset}}$	4	3	
$\overline{\text{Analyse}}$	6	5	
$\overline{\text{Error}}$	8	7	
GND	10	9	GND
$\overline{\text{SSReset}}$	12	11	
$\overline{\text{SSAnalyse}}$	14	13	$\overline{\text{Link0Speed}}$
$\overline{\text{SSError}}$	16	15	$\overline{\text{Link123Speed}}$
	18	17	
GND	20	19	GND

〔表11〕 サブ・トランスピュータの制御

信　号	リード/ライト	機械語アドレス
$\overline{\text{SSReset}}$	W	#80000000
$\overline{\text{SSAnalyse}}$	W	#80000004
$\overline{\text{SSError}}$	R	#80000000

に，インモス社により信号配置された 96 ピン DIN コネクタが使用されています.

サブ・トランスピュータを制御する $\overline{\text{SSReset}}$, $\overline{\text{SSAnalyse}}$, $\overline{\text{SSError}}$ の信号は，機械語アドレスで**表11** に示す番地に割り当てています. $\overline{\text{SSReset}}$, $\overline{\text{SSAnalyse}}$ は，最下位のビット 0 に“1”を書き込むことによりアクティブになり，“0”を書き込むことで非アクティブになります. $\overline{\text{SSError}}$ は最下位のビット 0 を読み出し，“1”であればエラーであることを示します.

▶その他の端子

LinkSpecial, MemWait, EventReq, MemReq, BootFromRom はスタティック RAM による構成と同じく GND に接続します.

●実装上の注意点

HB561008-150B ダイナミック RAM モジュールは，30 ピンのシングル・インライン・パッケージで，ソケットを必要とするソケット併用タイプ(リードレス・タ

イプ)のダイナミックRAMモジュールで，ソケットとしては山一のPS48を使用しました．

このダイナミックRAMモジュールは，8ビットのデータ幅なので，16ビットや32ビットのマイクロプロセッサを作成するのに便利ですが，技術の進歩は早いもので価格も安くならないうちに保守品種に指定するメーカも出てきました．

このタイプのモジュールは，1M×8タイプに移行しており，日立の製品ではHB56A18Bシリーズがあります．256K×8タイプから1M×8タイプのダイナミックRAMに変更することにより，マスタ・トランスピュータのメモリ容量は4Mバイトに拡張することができます．その場合には，MemAD12～20をDラッチで保持し，セレクタ74F257のA入力に加え，MemAD2～11をB入力に加えます．

ゲートICは，動作クロック周波数が20MHzですので高速タイプのTTL ICを使用する必要があり，アドレス・セレクタやラッチ，ストローブの各信号はFシリーズのTTLを使用しています．

3.5 回路のチェック

製作したトランスピュータ・ボードは，調整箇所がありませんので配線に間違いがなければ，すぐにTDS2を起動することができます．

回路のチェックは，まずリンク・アダプタの動作を確認します．リンク・アダプタのリンクの入出力どうしを接続し，出力をそのまま自身のリンク入力に加えて，書き込んだデータを読み出すことができることを確認します．

つぎに，トランスピュータ回路そのもののチェックを行います．最初はProcClockOutから20MHzのクロックが出力されていることを確認します．つぎにAnalyseが非アクティブな状態でResetをアクティブにし，一定時間の経過後リセットを非アクティブに戻します．

リセット後，内部コンフィギュレーションと外部コンフィギュレーション・サイクルが実行され，その間アドレスやデータ・バスの出力が変化するはずです．もし，外部メモリのタイミングにダイナミックRAMのサイクルを指定した場合には，定期的にリフレッシ

〔リスト1〕メモリ・チェック・プログラム

```
#include <stdio.h>
#include <conio.h>

#define TRUE  1
#define FALSE 0

/* リンク・アダプタ・アドレス */
#define LINKBASE       0xd0           /* リンク・ベース・アドレス  */
#define LINKREAD       LINKBASE       /* リンク・データ・リード   */
#define LINKWRITE      LINKBASE+2     /* リンク・データ・ライト   */
#define LINKINSTATUS   LINKBASE+4     /* リンク・ステータ・スリード */
#define LINKOUTSTATUS  LINKBASE+6     /* リンク・コマンド・ライト  */
#define LINKRESET      LINKBASE+8     /* リセット出力        */
#define LINKERROR      LINKBASE+8     /* エラー・フラグ入力     */
#define LINKANALYSE    LINKBASE+10    /* アナライズ出力       */
#define BIT_0          1              /* ビット・マスク       */

/* リセット，アナライズのタイミング作成 */
#define RESET_COUNT    10000          /* リセット出力タイミング  */
#define ANALYSE_COUNT  10000          /* アナライズ出力タイミング */
#define MAX_LINK_TIME  32000
#define LINK_COUNT     1000

/* メモリ・アクセスのためのヘッダ */
#define T_POKE         0              /* メモリの書込みヘッダ */
#define T_PEEK         1              /* メモリの読出しヘッダ */
#define MACHINEBASE    0x80000000     /* 機械語00アドレス   */


main()
{
      char ch;                        /* コマンド */
      int flag = 1;
      long  address;                  /* データ・アドレス */

      farstmess();                  /* オープニング・メッセージ */
      ResetT414();                  /* トランスピュータの初期化 */

      while ( flag == 1 ) {
          printf( "> " );
          scanf("%s %lx",&ch, &address );
          switch ( ch ) {
              case 'w':                     /* メモリの書込み */
              case 'W':
                  datapoke( address );
                  break;
              case 'r':                     /* メモリの読出し */
              case 'R':
                  datapeek( address );
                  break;
              default:
                  break;
          }
      }
}


/* オープニング・メッセージ */
farstmess()
{
      printf( "Transputer memory test program   ¥n¥n" );
      printf( "Data write command :  'W' or 'w'   ¥n" );
      printf( "> W address                        ¥n" );
      printf( "address: data                    ¥n¥n" );
      printf( "Data read command. :  'R' or 'r' . ¥n" );
      printf( "> R address                      ¥n¥n" );
}


/* トランスピュータのリセット */
ResetT414 ()
{
      int i;

      outp( LINKANALYSE, FALSE );         /* アナライズ解除 */
      outp( LINKRESET, FALSE );           /* リセット解除 */
      for ( i=0; i < RESET_COUNT; i++ )
      ;
      outp( LINKRESET, TRUE );            /* リセット */
      for ( i=0; i < RESET_COUNT; i++ )
      ;
      outp( LINKRESET, FALSE );           /* リセット解除 */
      for ( i=0; i < RESET_COUNT; i++ )
      ;
}


/* メモリへのデータ書込みコマンド */
datapoke( address )
 long address;
{
      long setdata;
      long faddress;

      faddress = address & 0xfffffffc;
      printf("%08lX: ",faddress);           /* アドレスの表示 */
      while( 0 != scanf( "%lx",&setdata )) {
                                            /* データ入力 */
         PokeTransputerWord( faddress, setdata );
                                            /* データ書込み */
```

ュが行われるのが確認できます.

メモリのテストは，**リスト1**のプログラムで行うことができます．これは，トランスピュータのメモリの内容をリンクを通して読出し/書込みを行ってテストするための簡単なプログラムで，MS-C ver 4.0 で書かれています．トランスピュータは，BootFromRom を GND に接続しリンクからのプログラムのブートを待っている間は，トランスピュータのメモリの内容を参照したり，書き換えたりすることができますが，この機能によってメモリの動作を確認することができます.

テストのためには，リンク・アダプタのシリアル・リンク入出力をテストするマスタあるいはサブ・トランスピュータのリンクに接続しますが，このときリンクのチャネルは0～3いずれでもかまいません．プログラムを実行するとコマンド入力待ちになりますので，読出しなら，

　＞R　アドレス ↵

と入力すると，そのアドレスから16番地後までのデータを表示します.

書込みなら，

　＞W　アドレス ↵

と入力すると，8桁のアドレスを表示しデータ入力を待ちます．データの入力は16進数以外の文字が入力されるまで繰り返されます.

このときアドレスもデータも16進で入力します．なお，このデータ読込みや書込みを行うアドレスは，機械語のアドレスではなく Occam のアドレスで入力を行います．したがって，アドレスとして0を入力すると，機械語の#80000000番地のデータがアクセスされます.

3.6　T800への拡張

T800は，T414とピン・コンパチブルなトランスピュータで，浮動小数点演算ユニットを内蔵しているため実数演算の処理速度が大幅に向上します.

機械語命令はT414の上位コンパチブルで，浮動小数点演算ユニットによる浮動小数点演算命令やブロック転送などの命令が追加され，逆に浮動小数点演算ユニットを内蔵したことにより浮動小数点演算のサポート命令が削除されています.

T800では，T414で使用されていなかった端子に新たに信号を割り当てていますが，トランスピュータ・

```
            faddress += 4;
            printf( "%081X: ",faddress);
        }
}


/* メモリからのデータ読出しコマンド */
datapeek( address )
 long int address;
{
        int  j;
        long i;
        long data;
        long faddress;

        faddress = address & 0xfffffffc;
        printf("¥n              " );
        for ( i = 0; i < 16; i+=4 )
           printf( "              %1X",(faddress + i) & 0xf);
        putchar( '¥n' );
        for (j =0; j < 0x80; j+=0x10 ) {
           printf( "%081X: ",faddress+j);
           for ( i =0; i<=12; i+=4 ) {
              printf( "   ");
              PeekTransputerWord( faddress+i+j );
           }
           putchar( '¥n' );
        }
}


/* トランスピュータからのメモリ・データ読出し */
PeekTransputerWord( address )
 long address;
{
        ByteToLink(T_PEEK);                         /* メモリ読込み */
        LongWordToLink(MACHINEBASE + address);
                                                    /* アドレス出力 */
        LongWordFromLink();                         /* データ入力  */
}


/* トランスピュータへのメモリ・データ書込み */
PokeTransputerWord ( address, data )
 long address;
 long data;
{
        ByteToLink( T_POKE );
        LongWordToLink( MACHINEBASE + address );
        LongWordToLink( data );
}


/* ロング・ワードのデータ書込み */
LongWordToLink (w)
 long int w;
{

        int i;

        for ( i = 0; i<4; i++ ) {
                ByteToLink( w & 0xff );
                w >>= 8;
        };
}


/* 1バイトのデータ書込み */
ByteToLink (ch)
 int ch;
{
        while ( !(inp(LINKOUTSTATUS) & BIT_0 ) != 0)
        ;
        outp(LINKWRITE, ch);
}


/* ロング・ワードのデータ読出し */
LongWordFromLink()
{
        int i;
        long result = 0;

        for ( i=0; i<32; i+=8 )
           result |= ((unsigned long)ByteFromLink() << i);
        printf( "%081X",result );    /* 読込みデータの表示 */
}


/* 1バイトのデータ読出し */
ByteFromLink ()
{
        while ( !( inp( LINKINSTATUS ) & BIT_0 ))
        ;
        return ( inp( LINKREAD ));
}
```

ボードは回路の変更をすることなく，そのままT800と差し替えることができます．**表12**はT414とT800の端子の相違点です．

T800で新たに定義された端子は，すべて入力です．ErrorInは，複数のトランスピュータを接続する場合，他のトランスピュータからのError出力を入力するために使用されます．

〔表12〕T800での新設ピン

ピン	T414の機能	T800の機能
B1	HoldToGND	ProcSpeedSelect0
D2	HoldToGND	ProcSpeedSelect2
D3	HoldToGND	ErrorIn
F1	HoldToGND	ProcSpeedSelect1

〔表13〕T800の動作クロックの選択

ProcSpeedSelect0～2	プロセッサ・クロック周波数(MHz)
0	20.0
1	22.5
2	25.0
3	30.0
4	35.0
5	－
6	17.5
7	－

ProcSpeedSelect0～2はプロセッサの動作クロック・スピードを選択するもので，**表13**のようにT800では17.5MHzから35MHzまで6種類のクロック・スピードを選択することができます．T414では，これらの入力はすべてHoldToGNDでGNDに接続されていますので，そのまま差し替えると，ErrorInを使用せず，動作クロックは20MHzという設定になります．

T800を使用する場合には，回路はそのままで使用できますが，追加された命令や削除された命令がありますので，Occamなどで記述されたプログラムはT800用に再コンパイルする必要があります．

4 トランスピュータの開発環境

トランスピュータの開発システムであるTDS2は，フォールディング(Folding)・エディタと呼ばれる独特のエディタをもっており，複数のプログラム・ステートメントを，折りたたんだり広げたりするようにし

コラム●Occamによる並行処理

リスト2で紹介したOccamによりマンデルブロ曲線を描くプログラムは，C言語で記述したプログラムをそのままOccamに書き直したもので，並列実行を行っている部分はありません．

このプログラムを並列実行させるためには，プログラム内から並列実行可能な部分を探し出し，並列実行を要求する必要があります．**リスト2**では，49～50行目の実部(変数r)と虚部(変数i)を求める部分が，並列実行できそうです．単独のトランスピュータならば，この二つのステートメントの上に，並行実行を要求する予約語“PAR”を付けるだけで並列実行を行うことができます．ただしこのプログラム例の場合，シングルプロセッサ上では並列実行をしても実行速度は速くなりません．

複数のトランスピュータで，並列に実行させるためのプログラムの最低単位は，予約語“PROC”によるプロシージャ単位となりますが，複数のトランスピュータで実行させるためには49～50行目の一方または両方の処理を抜き出し，並行に実行するプロシージャとして他のトランスピュータに割り当てる必要があります．

処理を分散することによって実行速度を向上することができますが，ここの例では処理ステップが1～2ステートメントと少なく，その割には引数の数が多くなってしまうので，それほど処理能力の向上効果は得られません．

いずれにせよ，マルチプロセッサ・システムの性能を最大限に引き出すためには，たんにシーケンシャルに考えたプログラムから並列実行可能な部分を探し出すのではなく，最初から並列実行を意識したアルゴリズムを考え出すことが重要です．マルチプロセッサ・システムではアルゴリズムにより実行速度にかなり大きな差が出てしまい，さらに各プロセッサの負荷量にばらつきがあるとマルチプロセッサ・システム全体がもつ最大のパフォーマンスを生かすことができません．

最後に，複数のトランスピュータにプロシージャを割り当てる方法について説明します．トランスピュータでは，複数のプロセッサで並列実行するためには，コンフィギュレーション(configuration)と呼ばれるプロセッサの接続状況とプロシージャの割当て情報を作り，おのおののプロセッサに処理を割り当てる必要があります．

リストAは，並列に実行する二つのプロシージャ例で，これらはputrの変数aaを10倍してからgetrに渡し，さらに10倍してputrの変数bbに返すという簡単なものです．

リストBは，これらputrとgetrを別々のプロセッサに割り当てるためのコンフィギュレーションの記述例で，プロセッサ0にputr，プロセッサ1にgetrを割り当てています．予約語PLACEは，チャネル型変数をどのリンクに割り当てるかを示すもので，数値はOccamマップ上のアドレス(リンクのアドレス)です．この**リストB**のプロシージャを実行するためには，ケーブルでリンクをつぎのように接続する必要があります．

プロセッサ0		プロセッサ1
Link1Output	⟷	Link0Input
Link2Input	⟷	Link1Output

て表示します．

折りたたんだところは，ブロックとして名前をつけシンボル化して表示することができます．この機能により，プログラムを階層的に表示し，プログラムの構造をわかりやすく表示することができます．

4.1 TDS2 のインストール

D700D バージョンでは，TDS2 のシステムなどが何枚かのフロッピ・ディスクに分けられ，圧縮されたファイル形式で格納されています．圧縮されているファイルは，ファイルの復元ユーティリティにより，圧縮される前のファイル形式に戻す必要があります．

TDS2 は基本的にはハード・ディスク上で動作するように考えられており，一連の変換はバッチ・ファイルにより自動的にハード・ディスク上にファイルが復元されます．元に復元されたファイルは最終的にユーティリティ，システム，ライブラリ，ソースなど，合わせて約 4 M バイトになりますが，変換する途中では最大 8 M バイトのディスク容量を必要とします．したがって変換するときには十分に余裕をもったハード・ディスクが必要で，もしハード・ディスクに余裕がない場合や，ハード・ディスクを使えない場合はファイルごとに変換します．

TDS2 のシステムはバッチ・ファイルの実行によりAドライブ(ハード・ディスク)上にディレクトリ"¥TDS2¥SYSTEM"が作成され，その中に格納されます．TDS2 を起動する前にはパス指定を"A:¥TDS2¥SYSTEM"に指定する必要があります．そして自分の作業用のディレクトリを作成し"¥TDS2¥SYSTEM"のディレクトリから"TOPLEVEL.TOP"と"TOPLEVEL.TKT"というファイルをコピーします．これで TDS2 を実行できる環境が整います．

このとき，リンク・アダプタのリンクとマスタ・トランスピュータのリンク 0 は接続されていなければなりません．

4.2 TDS2 の起動

MS-DOS 上で動作する TDS2 の D700D バージョンでは，PC-9801 用に NECTDS2 というバッチ・ファイルがあり，これで TDS2 を起動します．

TDS2 の起動は，

〔リスト A〕
並列実行させるプロシージャ

```
PROC putr( CHAN OF ANY comt01, comt02 )
  INT aa, bb:
  PAR                                   ……PAR 以下は並列実行させる
    SEQ
      aa := 30
      comt01 ! aa * 10                  ……変数 aa を 10 倍してリンクへ出力
    comt02 ? bb                         ……リンクから変数 bb への入力を待つ
:

PROC getr( CHAN OF ANY comt20, comt21 )
  INT a, b:
  SEQ
    comt20 ? a                          ……リンクから変数 a への入力を待つ
    comt21 ! a * 10                     ……変数 a を 10 倍してリンクへ出力
:
```

〔リスト B〕
コンフィギュレーション

```
CHAN OF ANY comt01, comt02, comt20, comt21:  ……チャネル型変数の宣言

PLACED PAR
  PROCESSOR 0 T4                       ……プロセッサ 0 は T414
    PLACE comt01 AT 1:                 ……Link1 出力
    PLACE comt02 AT 6:                 ……Link2 入力
    putr( comt01, comt02 )             ……プロセッサ 0 に割り当てるプロシージャ

  PROCESSOR 1 T4                       ……プロセッサ 1 は T414
    PLACE comt20 AT 4:                 ……Link0 入力
    PLACE comt21 AT 1:                 ……Link1 出力
    getr( comt20, comt21 )             ……プロセッサ 1 に割り当てるプロシージャ
```

A>NECTDS2 −t NEW.TOP

と入力するとTDS2が起動し，この例では“NEW.TOP”というフォールディング(Folding)が作られます．

フォールディングは，TDS2によりファイル作成などの作業をするための一種のディレクトリのようなものです．“−t”は，新たにフォールディング・ファイルを作ることを知らせるためのコマンド・オプションです．なお，フォールディング・ファイルのファイル属性は必ず，“.TOP”でなければなりません．

また，すでにフォールディング・ファイルが作られている場合は，

A>NECTDS2

と入力するだけでTDS2が起動します．

NECTDS2のバッチ・ファイル内では，起動時にトランスピュータのメモリ容量を1Mバイトに指定しています．今回製作したサブ・トランスピュータは，マスタとして動作させることも可能ですが，NECTDS2.BATの設定のままではメモリ容量が不足しますので，このままでは動作しません．

メモリ容量はバッチ・ファイルの中で“−S”オプションに続いて“#”から始まる16進数で指定します．メモリ容量を128Kバイトに指定するためには，バッチ・ファイルで指定しているメモリ容量“#100000”を“#20000”に書き換えます．ただし，TDS2は動作しますが，Occam言語コンパイラの大きさは128Kバイトを越えていますのでメモリ容量不足によりコンパイラは実行できません．

おわりに

回路図が書き上がったときのそっけなさ(もっとも英文のマニュアルには苦労しましたが)，そして回路が組み上がったときのちっぽけさ(32ビットのマイクロプロセッサの回路が，たかだか10cm四方の大きさに納まってしまった)に驚きました．そして，これらの回路はシリアル・リンクと数本の制御線で接続することによって，マルチプロセッサ・システムを構築することができるのです．

トランスピュータとは，トランジスタとコンピュータを掛け合わせた造語で，トランジスタのようにマイクロプロセッサをコンピュータ内でふんだんに使えるようにしたいと名付けたということですが，実際に回路を組んでみると，そのことがよくわかるような気がします．

トランスピュータの利点は，リンクからプログラムをロードすることができ，そしてプログラムをロードするためのローダ・プログラムが必要ないということです．マイクロプロセッサは8ビットから16ビット，そして32ビットとビット幅が広くなり高性能化してきました．ところがROMのデータ幅は8ビットのまま(ROMが16ビットや32ビットではお化けになってしまいますが)で，32ビット・プロセッサでは4個も必要になります．ROMが4個ともなるとROM焼き作業も大変で，きっと混乱してしまうことでしょう．それから，外部メモリ・アクセスのためのタイミングが可変で，ダイナミックRAMのリフレッシュ回路を内蔵しているため，ダイナミックRAMを使用してもスタティックRAMの場合と大差ない外付け回路で構成できました．

回路のコンパクトさだけに気を取られていましたが，やはり問題は実行速度です．相手としてはちょっと貧弱ですが，ハードウェアの最終チェックのつもりでマンデルブロ曲線を描かせるプログラムを実行してみました(**リスト2**)．

グラフィックス表示にはカノープス社のグラフィック・ドライバEGR98をエスケープ・シーケンス・モードで使用しています．MS-C ver 4.0で書いたプログラム(**リスト3**)を10MHzのV30(PC-9801VM21)により実行した場合と，T414(外部メモリ・サイクルは200ns)とOccamの組合せで実行し比較したところ，T414が約8倍から12倍の速度で実行することが確認できました．これが浮動小数点演算ユニットを内蔵したT800では，浮動小数点演算をT414の20倍程度の速さで実行できるということです．さらにこれらは並列接続できますから，そのパフォーマンスは計り知れません．

参考文献

1) INMOS, *The TRANSPUTER DATABOOK*, 1989.
2) INMOS, T800 Data Sheet, 1987.
3) INMOS(Pete Moore), Technical note 8, *IMS B010 NEC add-in board*, 1987.
4) INMOS(Tony Gore and David Cormie), Technical note 9, *Designing with the IMS T414 and IMS T800 memory interface*, 1987.
5) INMOS(Paul Walker), Technical note 29, *Dual Inline Transputer Modules (TRAMs)*, 1987.
6) INMOS, *The transputer instruction set — a compiler writers' guide*, 1987.
7) INMOS(Dick Poutain), *A tutorial introduction to Occam programming*, 1987.
8) INMOS, *TRANSPUTER DEVELOPMENT SYSTEM*, Prentice Hall, 1988.
9) 尾内理紀夫,『Occamとトランスピュータ』,共立出版,1986年.
10) 阿江　忠,「連載 マルチプロセッサ①」,『インターフェース』, 1985年1月号, pp.294〜303.
11) ピーター・C・モリス,「Transputerで身近に体験できる並列処理の世界」,『日経バイト』, 1987年7月号, pp.126〜136.
12) 渕上季代絵,『フラクタルCGコレクション』, サイエンス社, 1987年10月25日.

なすかわ・のりひろ　岩手大学工学部

〔リスト２〕マンデルブロ作図のプログラム(Occam)

```
001:  -- graphic example 2
002:  PROC apexe2( CHAIN OF ANY keyboard, screen ) -- プログラムの宣言
003:    #USE "¥tdsiolib¥userio.tsr"              -- ライブラリのインクルード
004:    VAL KL IS 30 :                      -- 最大繰り返し回数
005:    VAL KS IS 400 :                     -- 複素平面の縦横の分割数
006:    INT t, c, dummy :
007:    REAL32 rs, re:                      -- rs:複素平面の実部の始点  re:終点
008:    REAL32 is, ie:                      -- is:複素平面の虚部の始点  ie:終点
009:    REAL32 ar, ai:
010:    INT xx, yy :
011:    REAL32 zr, zi, zr1, zi1:            -- zr:複素変数(z)の実部  zi:虚部
012:    INT k :
013:    REAL32 dr, di:                      -- dr:実部の増分  di:虚部の増分
014:    REAL32 r, i, cr, ci:                -- cr:複素定数cの実部  ci:虚部
015:    BOOL ok :
016:    SEQ                                 -- SEQはプログラムの逐次(シーケンシャル)実行
017:      PROC esc()                        -- プロシージャescの宣言
018:        SEQ
019:          write.char( screen, #1B( BYTE ) )    -- コンソールへ1BH出力
020:          write.char( screen, 7( BYTE ) )
021:      :
022:      SEQ                               -- メイン・プログラム
023:        rs := -2.2( REAL32 )
024:        re := 0.5( REAL32 )
025:        is := -1.35( REAL32 )
026:        ie := 1.35( REAL32 )
027:        ar := 0.2( REAL32 )
028:        ai := 0.675( REAL32 )
029:        esc()
030:        write.text.line( screen, "INIT " )          -- EGR98の初期化
031:        esc()
032:        write.text.line( screen, "SCREEN 3,0 " )    -- 画面設定
033:        esc()
034:        write.text.line( screen, "CLS 3 " )         -- 画面クリヤ
035:        dr := ( re-rs )/400.0( REAL32 )
036:        di := ( ie-is )/400.0( REAL32 )
037:        cr := rs
038:        SEQ xx = 0 FOR 100                          -- 無条件ループ
039:          SEQ
040:            ci := is
041:            SEQ yy = 0 FOR 400                      -- 無条件ループ
042:              SEQ
043:                zr := 0.0( REAL32 )
044:                zi := 0.0( REAL32 )
045:                ok := FALSE
046:                k := 0
047:                WHILE ( k < 30 ) AND ( ok = FALSE )          -- 条件ループ
048:                  SEQ
049:                    r := ((zr*zr)-(zi*zi)) + cr             -- 実部の計算
050:                    i := ((2.0( REAL32 )*(zr*zi))) + ci     -- 虚部の計算
051:                    IF                                      -- 条件文
052:                      ((r*r)+(i*i)) > 4.0( REAL32 )  -- 複素数が2を越えたか
053:                        SEQ
054:                          c := k REM 7               -- 表示の色を求める
055:                          esc()                      -- 画面へドッドで表示
056:                          write.full.string( screen, "PSET " )
057:                          write.int( screen, xx, 0 )
058:                          write.char( screen, ',' )
059:                          write.int( screen, yy, 0 )
060:                          write.char( screen, ',' )
061:                          write.int( screen, c+1, 0 )
062:                          newline( screen )
063:                          ok := TRUE
064:                      TRUE                           -- 複素数の大きさが2を越えない
065:                        SEQ
066:                          zr := r
067:                          zi := i
068:                    k := k+1
069:                ci := ci + di
070:            cr := cr + dr
071:      keyboard ? dummy    -- キーボードの入力待ち(TDSにすぐ戻らないようにするため
072:  :
073:
```

〔リスト3〕マンデルブロ作図のプログラム(C言語)

```
001:  #include <stdio.h>
002:
003:  #define KL 30                       /*  最大繰り返し数  */
004:  #define KS 400                      /*  複素平面の縦横の分割数  */
005:
006:  main()
007:  {
008:
009:      int t, c;
010:      float rs =-2.2, re =0.5;    /*  rs: 複素平面の実部の始点  re: 終点  */
011:      float is=-1.37, ie =1.35;   /*  is: 複素平面の虚部の始点  ie: 終点  */
012:      float ar=1, ai= 0;
013:      int xx, yy;
014:      float zr, zi;               /*  zr: 複素変数 ( z ) の実部  zi:虚部  */
015:      float zr1, zi1;
016:      int k;
017:      float dr, di, r;            /*  dr:実部増分  di:虚部増分  */
018:      float i, cr, ci;            /*  cr:複素定数 c の実部  ci:虚部  */
019:
020:      putchar( 0xc );
021:      putchar( 0x7 );
022:      printf("¥033¥007 INIT¥n" );          /*  EGR98初期化  */
023:      printf("¥033¥007 SCREEN 3,0¥n" );    /*  画面設定  */
024:      printf("¥033¥007 CLS 3¥n" );         /*  画面クリヤ  */
025:      dr =( re-rs )/KS;
026:      di =  ( ie -is )/KS;
027:      cr = rs;
028:
029:      for ( xx = 0; xx < 400; xx++ ) {
030:          ci= is;
031:          for ( yy = 0; yy < 400; yy++ ) {
032:              zr = 0;
033:              zi = 0;
034:              for ( k = 1; k < KL; k++ ) {
035:                  r = (zr*zr) - (zi*zi) + cr;  /*  実部の計算  */
036:                  i = (2*zr*zi) + ci;          /*  虚部の計算  */
037:                  if (((r*r)+(i*i)) > 4 ) {    /*  複素数が2を越えたか  */
038:                      c = k % 7;               /*  表示の色を求める  */
039:                      /*  画面へドッドで表示  */
040:                      printf( "¥033¥007 PSET %d,%d,%d ¥n", xx, yy ,c+1 );
041:                      break;
042:                  }
043:                  else {                       /*複素数の大きさが2を越えない  */
044:                      zr = r;
045:                      zi = i;
046:                  }
047:              }
048:              ci = ci+di;
049:          }
050:          cr = cr+dr;
051:      }
052:  }
053:
```

グラフ理論の基礎と同形の概念

白川　功

点と点の関係だけに注目して，その間の関係やその間の流れを抽象化して把えるグラフ理論は，情報処理の分野においても重要な意味をもっている．たとえば回路をグラフとして表現し，回路の動作をシミュレートしたり，コンパイラのコード生成フェーズにおける最適化にグラフ理論を応用したりと，その応用範囲は広い．そこで，理論を実用化するという視点の下に，本連載を始める．まず第1回目の今回は，グラフ理論の基本的な概念を紹介した後，トポロジカルな意味での同形という概念および同形を判定する手法について解説する．　　　　　(編集部)

1　はじめに

本シリーズは，グラフの基礎概念を実用上の観点から説くことを目的とする．基礎という概念は人によって変動するのが世の常である．なおさらのことであるが，実用ということになると，千差万別である．何をもって実用というかは，人によってまちまちなのである．

そこで，筆者は，本誌の読者の興味の中心であると思われる，コンピュータおよびコンピュータ・ネットワークの設計・解析・運用，あるいはそれにまつわる各種プログラムの研究開発において，どのような話題が実用上もっとも重要な概念であるかを考慮して，本シリーズの執筆にとりかかることにする．

筆者も例にもれず，基礎および実用という観点からは，他の研究者と意を異にするであろう．しかしながら，筆者は長年にわたって，実際に実用に供するグラフ理論の研究開発に従事してきたので，その経験に基づいた実用的なグラフ理論を展開する資格があるのではないかと考えて，このシリーズの執筆をお引き受けした次第である．当然のことながら，ときには独断と偏見によって話題を選ぶかもしれないが，現実に実用上役立っているという基準のもとで議論を進めて行くことにする．

要するに，グラフ理論はこれほどまでに，広範な裾野をもつに至ったということであり，本シリーズで展開される話題は，グラフ理論のなかでもごく一部であるということを忘れてはならない．

1.1 グラフの定義

グラフ(graph)は，いくつかの**点**(node または vertex)とそれらの点の対を両端とする線分 ── これを**枝**(edge または branch)と呼ぶ ── からなる．グラフでは，各枝がどの点とどの点を両端点にもつか，すなわちどの点とどの点の間に**つながっているか**(あるいは**接続しているか**)だけを問題とし，点の位置とか枝の形状とかを問題としない．以下では，与えられたグラフ G の点集合が V，枝集合が E であるとき，G を $G=(V,E)$で表す．とくに断らないかぎり，V の各元をアルファベットの小文字で，E の各元を数字で表記する．

グラフ $G=(V,E)$が与えられるということは，点の集合 V が表示され，かつ各枝 $k\in E$ がどの点とどの点の間につながっているかの接続情報が何らかの形で明示されているということを意味する．

1.2 無向グラフ

グラフの接続構造を問題にするとき，各枝の両端点への接続において向きを考慮に入れる場合と入れない場合とがある．与えられたグラフ G のすべての枝がその両端点への接続において向きをもたないとき，そのグラフ G を**無向グラフ**(undirected graph)といい，すべての枝がその両端点への接続において向きをもつとき，G を**有向グラフ**(directed graph)という．

【例 1.1】　図 1.1(a)の線図形 G において，4個の点からなる点集合 $V=\{a,b,c,d\}$ と7個の枝からなる枝

集合 $E=\{1,2,\cdots,7\}$ に注目する．各枝 $k=1,2,\cdots,7$ がどの点とどの点につながるかの接続情報が図によって明示されているから，この線図形は一つの無向グラフ $G=(V,E)$ を表す．

このグラフの各枝がつながる両端点を表(以下では**接続表**と呼ぼう)で表すと，**図1.1**(b)のようになる．この接続表から同図(a)の線図形 G が再現可能であるから，グラフ G を**図1.1**(a)の線図形で描くことと，同図(b)の接続表で与えるということは同等である．すなわち，グラフの接続構造を**図1.1**(a)の図形で与えても，あるいは同図(b)の接続表で与えても，本質的には同等である．□

上の例題で述べたように，グラフの接続構造を，通常は線図形として表すか，あるいは接続表で表す．ただし，コンピュータ実行の観点からは，目的に応じてグラフを表示する適当なデータ構造を整えなければならないことはいうまでもない(これについてはグラフの算法の項で言及する)．

無向グラフにおいて，枝 k が点 u と点 v の間につながるとき，非順序対(すなわち順序が付かない対) $k=\{u,v\}$ で表し，点 u,v は互いに**隣接する**(adjacent)という．たとえば，$1=\{a,b\}$，$2=\{a,c\}$ であり，点 b,c は点 a に隣接している．両端点が同一の点であるような枝を**自己閉路**(self-loop)といい，両端点の対が互いに同一であるような枝を**多重枝**(multiple edges)という．たとえば，**図1.1**(a)の枝 $7=\{a,a\}$ は自己閉路であり，枝 $4=\{b,d\}$，$5=\{b,d\}$ は多重枝をなす．

1.3 有向グラフ

有向グラフは，無向グラフの各枝に「向き」という付加的な属性が加わっただけである．枝 k が点 u から出て，点 v に入るとき，点 u および点 v をそれぞれ枝 k の**始点**(initial vertex)および**終点**(terminal vertex)といい，**順序対**(対の要素に順序が付けられたもの) $k=(u,v)$ で表し，点 u,v は互いに隣接するという．なお，有向グラフの各枝の接続の向きを表示する場合には，線図形においては矢印で，接続表においては(始点，終点)の順序対で表す．

無向グラフと同様に，同一点を両端点にもつ枝を自己閉路といい，両端点の対が同一である枝を多重枝という．無向または有向グラフ G が自己閉路，多重枝のいずれをももたないとき，G は**単純**(simple)であるという．

【例 1.2】 **図1.2**(a)で描かれる線図形において，各枝の向きを含めた接続情報が明示されているから，これは一つの有向グラフを表す．このグラフ G の接続表を同図(b)に示す．枝 $1=(a,a)$ は自己閉路であり，枝 $4=(b,c)$，$5=(b,c)$ は多重枝である．□

1.4 グラフ理論の工学への応用

グラフとは，このようにいくつかの点の集合とそれらの間につながるいくつかの枝の集合とからなる，きわめて単純な構造をもつ**組合せ論的**(combinatorial)な対象物であり，**グラフ理論**(graph theory)とは，このような点と枝との間の接続構造にかかわる諸性質を究明するための理論である．通常，数学における『xx理論』といえば，それはいつどこで誕生したかが断定できないことが多い．しかし，このグラフ理論は珍しくはっきりしており，かの有名な「Königsberg の橋の問題」と呼ばれる一種のパズルの問題に関するスイスの数学者 L. Euler(1707-1783)の論文(1736)に始まるとされている[1]．以来，その構造上の単純性のために工学に限らず，理学，医学，社会学，経済学，など広い分野にわたって応用されてきた．

1970 年のマイクロプロセッサの発明以来，VLSI 技術の著しい技術革新に支えられて，コンピュータを取り巻く情報処理技術が飛躍的に進展しているが，これにともなってエンジニアリング・ワークステーション(EWS)の高性能化，高機能化が強力に推進され，その進展普及には目を見張るものがある．このような研究開発の環境の急激な変貌とともに，コンピュータ援用設計(CAD)，コンピュータ援用製造(CAM)，コンピュータ統合製造(CIM)，工場自動化(FA)，あるいはオフィス・オートメーション(OA)の各種手法の開発のピッ

〔図1.1〕 無向グラフ G

(a) 線図形

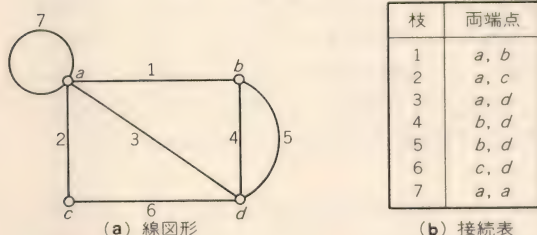
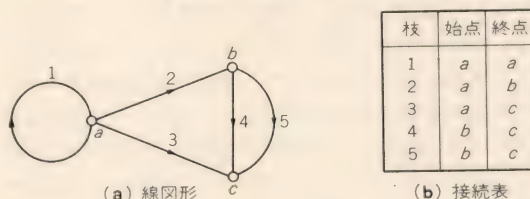
枝	両端点
1	a, b
2	a, c
3	a, d
4	b, d
5	b, d
6	c, d
7	a, a

(b) 接続表

〔図1.2〕 有向グラフ G

(a) 線図形

枝	始点	終点
1	a	a
2	a	b
3	a	c
4	b	c
5	b	c

(b) 接続表

チがさらに急速に上昇しようとしているが，近年の各種システムに対する設計・制御・運用の複雑度がさらに指数的に増大しつつあるという趨勢を見るにつけ，つねにこれらシステムの設計・制御・運用における品質向上という課題がつきまとう．設計・制御・運用の品質向上は，それにおける全過程の階層化，先鋭化，統合化がもっとも重要な鍵となる．このなかでも，とくに数理的思考の展開を必要とするのは，先鋭化の研究開発，すなわち設計・制御・運用の性能・機能向上のための**算法**(アルゴリズム，algorithm)あるいは**手続き**(procedure)の研究開発においてである．とくに，大規模システムの設計・制御・運用に付随する各種算法・手続きの研究開発においては，組合せ理論，とりわけグラフ理論は，きわめて有用な実用理論であり，今や工学における必要不可欠の基礎理論の一つともなっている．

本シリーズは，これら工学的応用という観点に立って，グラフに関する基礎的概念とその具体的な実用事例について概説を試みる．その際，とくにコンピュータ実行という観点を重視して議論を展開する．

2 グラフの同形とその応用

まず，グラフのもっとも基本的な概念としての同形という概念，およびそれの一つの応用としての同形判定の手法について考察する．

2.1 グラフの同形

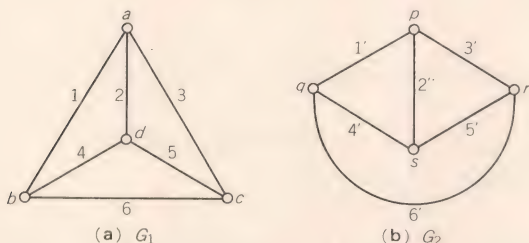
二つのグラフ $G_1=(V_1, E_1)$，$G_2=(V_2, E_2)$ が同数の点と同数の枝をもち，V_1 と V_2 との間および E_1 と E_2 との間にそれぞれ適当な1対1対応が存在して，その対応のもとで両者の接続関係が等しくなるとき，G_1 と G_2 とは**同形である**(isomorphic)であるという．言い換えれば，1対1対応(全単射)

$$f : V_1 \to V_2$$

で，$\{u, v\} \in E_1$ であるとき，かつそのときにかぎり $\{f(u), f(v)\} \in E_2$ であるような f が存在するとき，G_1 と G_2 は同形であるといい，f を**同形写像**という．

〔図2.1〕同形であるグラフの例

(a) G_1

(b) G_2

【例2.1】 **図2.1**のグラフ G_1，G_2 の点集合の間の1対1対応 f；

$$f(a)=p,\ f(b)=q,\ f(c)=r,\ f(d)=s,$$

および枝集合の間の1対1対応 g；

$$g(k)=k' \quad (k=1, 2, \cdots, 6)$$

に注目すれば，G_1 の各枝 $k=(u, v)$ に対応する G_2 の枝 $k'=g(k)$ は $k'=(f(u), f(v))$ で表されるから，G_1 と G_2 とは同形であり，f は同形写像である．□

直感的にいえば，G_1 と G_2 とが同形であるということは，G_1 の点と枝の記号や番号の付け方を適当に変えると G_2 が得られるということ，すなわちグラフの描き方が同じであるということを意味する．

2.2 グラフの同形判定とその応用

与えられた二つのグラフが同形であるかどうかを判定するアルゴリズムはVLSIの接続検証に有用な応用をもつ．これについて，以下に述べる．

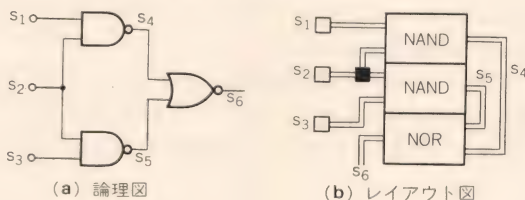
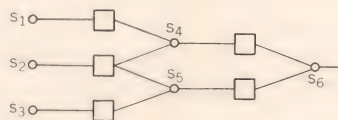
いま，**図2.2**(a)で示されるような論理図に対してレイアウト設計を行った結果，同図(b)のレイアウト図が得られたとする．この場合，レイアウト設計が正しく行われているかどうか，すなわちレイアウト設計における各機能セルが論理図どおりに結線されているかどうかが問題となる．換言すれば，論理設計の結果を蓄えるデータ・ファイルから抽出して得られる機能セル・信号間の結線を表すグラフ(たとえば，**図2.3**参照)と，レイアウト設計の結果を蓄えるデータ・ファイルから得られる機能セル・信号間の結線を表すグラフとが同形かどうかが問題となるのである．

この問題に対して，実用上有用なアルゴリズムが考案された．以下に，その概要を述べる．

集合 A の互いに素な二つの部分集合 A_1，A_2 $(A_1 \cup$

〔図2.2〕論理図とレイアウト図

〔図2.3〕論理図に対応する結線図

〔図2.4〕同形判定アルゴリズム

```
Algorithm ISOMORPHISM

[INPUT] A graph G=G¹+G².
[OUTPUT] An answer
         either "isomorphic" or "nonisomorphic".

begin
  procedure BACKTRACK(π, i, {u, v}) begin
    comment u∈V¹, v∈V², and the partition π is
            denoted by {V[1],..., V[i],..., V[p]};
1   π←{V[1],..., V[i]−{u, v},..., V[p], {u, v}};
2   refine the partition π;
3   if there exists a block V[j]=Vj¹+Vj² such that
      |Vj¹|≠|Vj²| then return;
4   if there does not exist any block V[j]=Vj¹+Vj²
      such that |Vj¹|=|Vj²|≥2 then goto isomorphic;
5   let V[k]=Vk¹+Vk² be a block such that
      |Vk¹|=|Vk²|≥2, and x be a vertex in Vk¹;
6   for each y∈Vk²
      do BACKTRACK(π, k, {x, y}); end;
7  π←{V¹∪V²};
8  BACKTRACK(π, 1, φ);
9  goto nonisomorphic;
end;
```

$A_2=A$, $A_1\cap A_2=\phi$)への直和分割を $A=A_1+A_2$ と書く．互いに素な二つのグラフ $G_1=(V_1,E_1)$，$G_2=(V_2,E_2)$ ($V_1\cap V_2=\phi, E_1\cap E_2=\phi$)に対して定義される**和グラフ** $G=(V_1\cup V_2,\ E_1\cup E_2)$ を $G=G_1+G_2$ で表す．

与えられた二つのグラフ $G^1=(V^1,E^1)$，$G^2=(V^2,E^2)$ が互いに同形であるということは，一つの同形写像 f；$V^1\to V^2$ が存在するということであるが，この f はグラフ $G=G^1+G^2$ 一つの**自己同形写像** f：$V\to V$ とみることもできる．したがって，G^1，G^2 の同形判定問題は，$G=G^1+G^2$ における自己同形写像の存否の判定問題に帰着される．

グラフ $G=G^1+G^2$ の点集合 $V=V^1+V^2$ の互いに素な部分集合 $V_1, V_2, \cdots, V_p$ への直和分割 π を $\pi=\{V_1, V_2, \cdots, V_p\}$ で表し，その各要素 V_i を分割 π のブロックと呼ぶ．各ブロック V_i は

$$V_i=V_i^1+V_i^2(V_i^1\subset V^1,\quad V_i^2\subset V^2)$$

のように細分される．

グラフ G のある分割 π の相異なる二つのブロック V_i，V_j に属するどのような2点 $u\in V_i$，$v\in V_j$ に対しても，$f(u)=v$ となるような自己同形写像 f が存在しないとき，分割 π は G における一つの**同形分割**という[2]．同形分割は，つぎの有用な性質をもつ．

〔図2.5〕細分化のアルゴリズム

```
Algorithm PARTITION

[INPUT] A graph G=[V, E], represented by
        adjacency lists Γ(v) for all v∈V, and
        an initial partition π={V[1], V[2],..., V[p]}.

[OUTPUT] A partition π*={V*[1], V*[2],..., V*[q]}
which satisfies the conditions C1, C2, and C3.
This algorithm omits certain implementation details.

begin
1  WAITING←{1, 2,..., p};
2  q←p;
3  while WAITING is not empty do begin
4    select and delete any integer i from WAITING;
5    INVERSE←{u|u∈Γ(v), v∈V[i]};
6    for each u∈INVERSE do adj(u)←|Γ(u)∩V[i]|;
7    for each j such that V[j]∩INVERSE≠φ do begin
8      make Q and each BUCKET empty;
9      for each u∈V[j]∩INVERSE
         do add u to BUCKET[adj(u)];
10     for each nonempty BUCKET[k] do begin
11       q←q+1;
12       create a new block V[q];
13       Q←Q∪{q};
14       V[q]←BUCKET[k];
15       V[j]←V[j]−V[q]; end;

16     if V[j]=φ then begin
17       V[j]←V[q];
18       eliminate the block V[q];
19       Q←Q−{q};
20       q←q−1; end;

21     let V[qm] be a block of maximum size
         in {V[qk]|qk∈Q∪{j}};
22     interchange V[j] and V[qm];
       comment in this stage,
         |V[j]| ≥ |V[qm]| and j∉Q;
23     add Q to WAITING; end; end;
 end;
```

［性質1］ グラフ $G=G^1+G^2$ の一つの同形分割 π のどれかのブロック $V_i=V_i^1+V_i^2$ で，

$$|V_i^1|\subset|V_i^2|$$

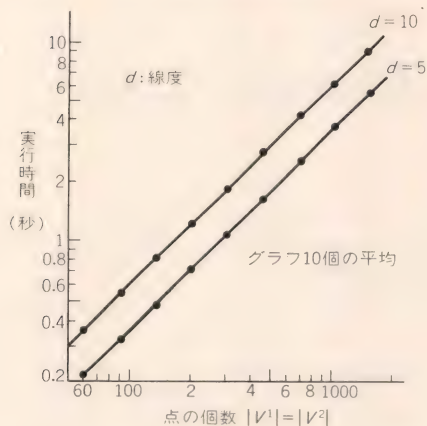
であるようなものが存在すれば，G^1 と G^2 は同形ではない．□

グラフ G の点 v に隣接する点の集合を $\Gamma(v)$ で表す．

［性質2］ グラフ G の同形分割 π のあるブロック V_j に含まれる2点 u, v に対して

$$|\Gamma(u)\cap V_i|\subset|\Gamma(v)\cap V_i|$$

であるようなブロック V_i が存在するとき，G の自己

〔図2.6〕アルゴリズムの実行時間(同形の場合)

同形写像 f で, $f(u)=v$ とするようなものはない. □

このような二つの性質を利用して, グラフ $G=G^1+G^2$ の自己同形写像の存否のための効率的な判定アルゴリズムが図2.4 のように構築される[3]. ただし, このアルゴリズムの行 2 の細分化のための手続きを図2.5 に示す. これは, Hopcroft[4]の状態数最小化のアルゴリズムに基づいている.

なお, 図2.5 において分割 π^* に対する条件 C1, C2, C3 とは, つぎのようなものである.

C1 : どのブロック $V_i^* \in \pi^*$ も, どれかのブロック $V_j \in \pi$ の部分集合である.

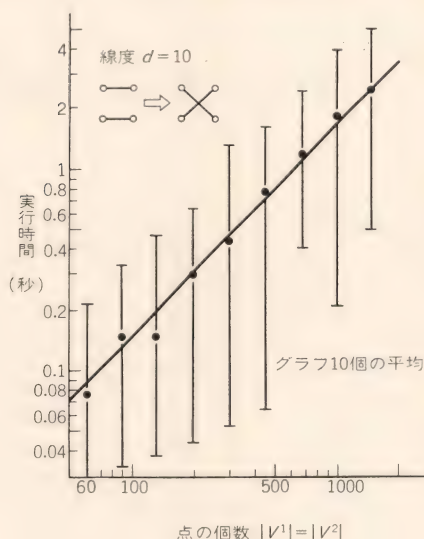
C2 : 同一ブロックに含まれるどの 2 点 u, v に対しても, 各ブロック $V_i \in \pi^*$ は

$$|\Gamma(u) \cap V_i^*| = |\Gamma(v) \cap V_i^*|$$

を満たす.

C3 : 分割 π^* は C1, C2 を満たすすべての分割のなかで, ブロックの個数が最小である.

図2.6 にこのアルゴリズムの実験結果を示す. 用いたグラフは同形判定のもっとも困難な**正規グラフ**, すなわち各点の**線度** d (各点につながる枝の個数)が同一であるようなグラフをランダム発生させたものである. 図からわかるように, グラフの点の個数に比例する実行時間で同形判定ができる. なお, 実験に使用したコンピュータは 1 MIPS 程度の汎用機である.

〔図2.7〕アルゴリズムの実行時間(同形でない場合)

正規グラフ G^1 に同形でないグラフとして, 適当な二つの枝を付け変えて得られるグラフ G^2 をつくり, 非同形判定の実験をも行った. 図2.7 はその一部を示す. 実行時間は大きくばらついているが, 最悪の場合でも同形判定の実行時間を越えることはなかった.

このアルゴリズムに基づいて作成されたプログラムは現在でも VLSI の検証に使用されている.

参考文献

1) N.L.Biggs, E.K.LLoyd, R.J.Wilson, *Graph Theory 1736-1936*, Clarendon Press, Oxford, 1976.
2) 榎本, 片山, 米崎,「分割アルゴリズムに基づく同型グラフの検索について」,『情報処理学会誌』, 18, 12, pp.1209-1217, 1977.
3) 久保, 白川, 尾崎,「グラフ間の同形判定アルゴリズムにおける効率化について」,『電子通信学会論文誌』, 61-A, 11, pp. 1099-1105, 1978.
4) J. Hopcroft, "An *n* log *n* algorithm for minimizing states in a finite automaton", *Theory of Machines and Computations*, Z. Kohavi and A. Paz, eds., Academic Press, pp. 189-196, 1971.

しらかわ・いさお 大阪大学工学部電子工学科

アメリカのインチキ性と Common Lisp

NHKの特派員をされていた日高義樹さんの本に『日本は「2番」でいい！』という題名の本がある．なかなか人には真似のできない素敵な題名であるが，それにもまして僕が感動を覚えたのは，この本の中で何回となく繰り返されている「インチキ性」という言葉である．もちろん，「インチキ性」とは，裏と表の二面性，本音と建前の二面性のことで，「そんなのインチキだっ」のインチキのことである．つまり，この本は，アメリカの持つ恐るべき「インチキ性」について詳しく解説し，そのような「インチキ性」を持っていない日本が一番になれるはずがないのだから，二番に甘んずるべきだと主張しているのである．

アメリカのインチキ性はいたるところにある．表では自由貿易を唱えて日本に農産物市場の開放を求める議員たちが，裏ではアメリカの農民の利益を代表していたり，表ではバードン・シェアリングを唱えて日本に防衛努力を迫る議員たちが，裏ではアメリカの軍事産業の黒幕であったりする．つまり，ここでいうインチキ性とは，誰にも文句のつけられないような正義の裏側に，自分勝手な欲望が潜んでいるということを意味する．そして，最近になって日本人もようやくアメリカの持つインチキ性に気がつき始め，ひどく腹を立てているようである．

現在では Common Lisp が LISP 言語の標準として定着したようである．Common Lisp 以前は，MacLisp, ZetaLisp, Franz Lisp, Interlisp, Standard Lisp などの数多くのLISP方言が存在し，AIなどの分野の LISP プログラマたちは，LISP 方言間のコンパチビリティのなさに悩まされ続けていた．たとえば，Franz Lisp と ZetaLisp は似てはいるが，ZetaLisp で開発されたプログラムを Franz Lisp に移植するのは一筋なわではいかない．ということで，Common Lisp は LISP プログラマの救世主のようにして登場した感があるが，実は裏には，LUCID 社に代表される LISP の商用化と，そこから得られる膨大な利益が見込まれていたのである．そして，そのためには，アメリカ主導で LISP 言語を標準化したほうがいいに決まっている．日本人はありがたがって Common Lisp の動きにのっかってしまったが，ずるがしこいヨーロッパ人たちは，なんやかんやと文句をつけて，アメリカに主導権を渡すまいとしている．

アメリカの持つ強力な二面性に対して，日本は表も裏もごちゃごちゃである．似たもの同士の社会の中で互いに腹の中までわかりあっているから，建前と本音を区別することが不可能であるか，意味を持たなかったからであろう．

しかし，世界は正義の名の下でしか動かない．だから，アメリカに対して「お前はインチキだ」といっても始まらないのである．さいわいにして正義は一つとは限らない．自分にもっとも都合のよい正義を選んで，それをあたかも唯一の正義のようにして人に押し付ける，というのが外交の基本である．

『日本は「2番」でいい！』に，アメリカの学校のディスカッションのクラスのことが書いてある．特定の問題（たとえば軍備拡張か軍縮か）に関して，自分が本当はどう思っているのかとはまったく無関係に，賛成と反対に分かれて議論をし合うという，要するに，屁理屈をこねあう練習である．そして，そのようにして幼い頃から屁理屈で鍛えられたアメリカ人に，日本人がかなうわけがないという．

しかし，である．正義は一つとは限らないとはいっても，正義の名の下に0と1をひっくり返すことはできない．そして，アメリカの唱える正義が戦後の世界を形作ってきたことは事実であるし，それはこれからも続くことではないだろうか．中国で民主化運動が流血の下に弾圧されているのを見れば，いかに我々日本人が自由を満喫しているかがわかるが，その自由の大部分はアメリカの唱える正義の名の下に与えられたものである．また，世界は今や「米ソ冷戦」の時代を終え，新たな世界秩序へ向かおうとしているが，僕は，40年もの間，冷戦が冷戦であり続けたということに感動を覚えずにはいられない．それはやはり，アメリカ（とソ連）が世界平和という正義を貫いたからこそだろう．

また，Common Lisp のお蔭で，どれだけ LISP プログラムのコンパチビリティが向上したことだろうか．今や，大型機でもワークステーションでもパソコンでも，同一のプログラムがなんの変更もなく動くのである．

僕は，アメリカのインチキ性はインチキ性として認識しつつも，インチキ性の表の建前の部分は裏の本音の部分とは別にして，素直に耳を傾けるべきではないだろうか，と最近になって思っている．人種差別反対，人権擁護など，アメリカが世界に押し付けてきた価値観のいくつかは，きわめて不偏的なものであり，我々日本人も大いに恩恵を受けているのである．

これに対して，アメリカのインチキ性そのものを学ぶべきかどうかはよく分からない．日本がアメリカのように豊かになり，それを維持していくためには，日本もある程度のインチキ性は学ばなくてはならないだろう．しかし，豊かさと幸福とは別である．アメリカのインチキ性は，あきらかに，日本人のメンタリティに反することなのではないだろうか．インチキをインチキと知りつつ主張する心が，けっして健康なものだとは思えない．アメリカ人が豊かさの中で本当に幸福なのかは，よく分からない．それが証拠に，アメリカにいる日本人研究者たちの多くは，インチキ性が必要になる年になると日本に帰って来る．

M 生

AIプログラミング言語：Prolog編(下)

田中 裕一／海野　敏

これまで，2回にわたってPrologを取り上げ，そのデータ，基本的な動作，プログラミングの基本技法について説明しました．今回はまとめとして，Prologの応用プログラムをいろいろ示し，それを通じて重要なプログラミング技法を解説します．

バックトラック

▶実行過程の表現 —— 導出木

図1は，前回紹介した家系図データベースのプログラムの一部です．いま，このプログラムに，

?- ancestor(ヒカルゲンジ，ニオウノミヤ).

というゴールを与えると，

yes

が返ってくることは，もうおわかりでしょう．このときの実行過程は，**図2**のように木のかたちで表現できます．そこで，この木のことを**導出木**と呼びます．

▶実行過程を追跡する

復習を兼ねて，実行過程を途中まで追跡してみましょう．まず，述語ancestorの第1節の頭部で

X＝ヒカルゲンジ

Y＝ニオウノミヤ

という代入によって単一化が行われ，導出の結果，ゴール節は，

?- parent(ヒカルゲンジ，ニオウノミヤ).

に変わります(①)．つぎに，述語parentの第1節を用いて導出を行い，

?- father(ヒカルゲンジ，ニオウノミヤ).

がゴールになります(②)．しかし，このゴールは失敗するのでプログラムの制御は後戻りし(③)，再び

?- parent(ヒカルゲンジ，ニオウノミヤ).

が試みられます．このような失敗による後戻りを**バックトラック**というのでした．

今度は，述語parentの第2節が試みられ，

?- mother(ヒカルゲンジ，ニオウノミヤ).

がゴールになり(④)，これも失敗します．プログラムの制御は後戻りしますが(⑤)，もはや述語parentに試みるべき選択肢はありません．そこでプログラムはさらに後戻りし(⑥)，

?- ancestor(ヒカルゲンジ，ニオウノミヤ).

が再度試みられます．

つぎに選ばれる選択肢は述語ancestorの第2節で，

?- parent(ヒカルゲンジ，Z),
   ancestor(Z，ニオウノミヤ).

が新しいゴールとなります．そしてまずparent(ヒカルゲンジ，Z)を試み(⑦)，さらにparentがfather(ヒカルゲンジ，Z)を呼び出します(⑧)．これは

Z＝レイゼイイン

という代入によって成功することがわかります．

この先は省略しますが，**図2**の矢印をたどっていくと，図の右下隅でたしかに解が得られていることがわかると思います．このようにして，最初の解が見つかるか，あるいは選択肢を試み尽くしてしまうまで，バックトラックを繰り返して探索が進められます．

迷路の脱出法に「右手法」というのがあります．こ

〔図1〕家系図データベース

```
/* facts */
father(ヒカルゲンジ,レイゼイイン).
mother(フジツボノミヤ,レイゼイイン).
father(ヒカルゲンジ,アカシノチュウグウ).
mother(アカシノウエ,アカシノチュウグウ).
father(キンジョウ,ニオウノミヤ).
mother(アカシノチュウグウ,ニオウノミヤ).

/* rules */
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

れは，右手を壁につけてひたすら壁に沿って進むと，ループがないかぎり，いつか必ず出口にたどり着くことができるというものです．Prologのプログラムの実行も，まさにこの右手法と同じことをやっているわけです．

▶ Prologの実行過程のまとめ

Prologの実行過程をまとめると，つぎのようになります．

(1) ゴール(一般にはサブゴールの列)が与えられると，最初のサブゴールから順に処理を行う．
(2) 一つのサブゴールに対し，それと同じ述語を頭部にもつプログラム節を捜し，それとの間で導出を試みる．
(3) そのようなプログラム節が複数個あるときには，先に書いてある節から順に組み合わせてみる．
(4) 導出に失敗したときには，バックトラックしてつぎのプログラム節を試す．

ここで(1)の意味するところは，最初のサブゴールを達成するまでは，つぎに進むことができないということです．サブゴールは導出によって，新しいゴールに書き換えられますが，さらにその先頭のサブゴール(サブサブゴール)を導出の対象として先に進みます．このように，実行は導出木の下へ下へと進んでいくので，この実行順序を“**depth-first**”といいます．

(3)と(4)では，プログラムの動きが必ずしも決定的でないことを述べています．つまり，実際に動かしてみるまで，プログラムの実行順序が決まらないわけです．このような性質を**非決定性**(non-determinism)といいます．

カット

このように，バックトラックというのはPrologの基本的な制御メカニズムになっています．しかし，ときにはバックトラックする必要がない場合，あるいはバックトラックしてはならない場合が生じます．このようなとき，バックトラックを抑制する方法が**カット**(cut)です．

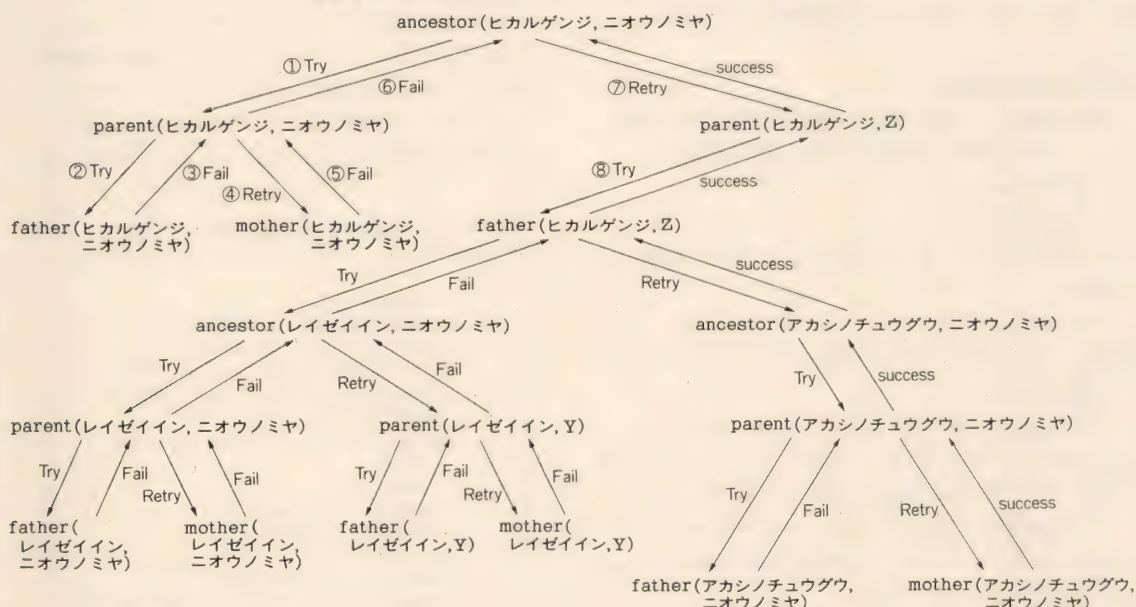
▶整数の絶対値をとる述語absの例

整数の絶対値を取る述語absを考えてみましょう．

```
abs(X,Y) :- X < 0, Y is -X.
abs(X,X) :- X >= 0.
```

というプログラムでは，Xが負のときは第1節が取られて，YにはXの符号を変えたものが入ります．そうでないときは，X<0が失敗して第2節に行き，X>=0ですから，Xの値そのものが絶対値となります．

すぐわかるように，X>=0という条件は冗長です．第2節にくるのは，第1節のX<0が失敗したときだ

〔図2〕ancestor(ヒカルゲンジ，ニオウノミヤ)の導出

けですから，そのときX>=0であることは明らかだからです．そこで，上のプログラムは

```
abs(X,Y) :- X < 0, Y is -X.
abs(X,X).
```

と書き換えてもよさそうです．

しかし，ちょっと待ってください．このabsが

```
...
repeat,
read(X),
abs(X,Y),
Y < 5,
...
```

という文脈の中で使われたとしたらどうでしょう．これは，Xにいろいろな整数値を入力してみて，そのうちでXの絶対値が5より小さいものだけを取り出すプログラムの一部です．

たとえば，Xに−7が入力されたとしましょう．absの第1節によりYは7となりますが，これはY<5の条件に合わないので，バックトラックします．すると，

〔図3〕カット・オペレータの使用

```
/* rules */
parent(X,Y) :- !,father(X,Y).
parent(X,Y) :- mother(X,Y).
```

今度はabsの第2節が選ばれてYは−7となってしまいます．これは間違いです．

▶カット・オペレータ“！”を使う

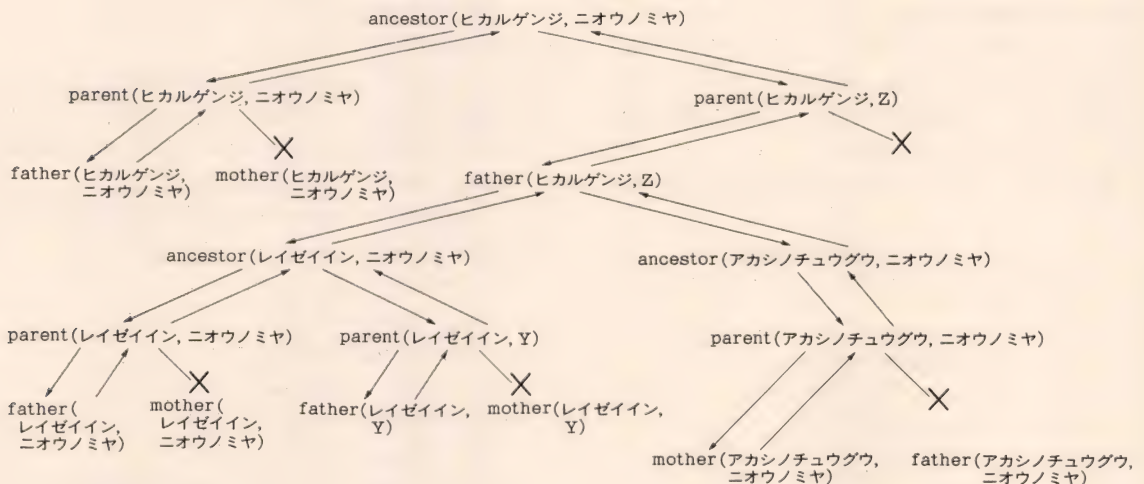
このようなことを防ぐためには，absの第1節が選ばれた時点で，それ以外の選択肢をカットしてしまえばよいのです．プログラム中でカットを実現するにはカット・オペレータ“!”を使います．これを使って，

```
abs(X,Y) :- X < 0, !, Y is -X.
abs(X,X).
```

のように書くことにより，X<0の条件が満たされた時点で，第2節は捨てられてしまいます．つまり，ここを通過した場合には，バックトラックしても，もはや他の選択肢は残されていないのです．

```
a :- b1,!,c1,d1.
a :- b2,!,c2,d2.
a :- b3,!,c3,d3.
a :- b4,!,c4,d4.
a :- e.
```

というプログラムの実行中に，d2で失敗が生じたとき，カット・オペレータがなければバックトラックしてc2を再度試み，それでも失敗したならb2を再度試み，それでもだめならaの次の節へ進むはずです．しかし，b2とc2のあいだにカット・オペレータがあるので，実行の制御はこの位置より前には戻りません．すなわち，b2の再試行およびaのつぎ以降の節の実行の可能性は，すでに切り捨てられてしまっているのです．

〔図4〕カット・オペレータの効果

カットは，導出木で考えると，それより右の枝を切り落としてしまうことに相当します．たとえば，家系図データベースのプログラムに**図3**のようにカット・オペレータを挿入した場合を考えてみます．こうすると，parent の第1節で失敗してもカットオペレータがあるために第2節に進めません．これは，**図4**のように導出木の枝を切り落としてしまうことと同等です．

▶どういうときにカットを使うか

実際のプログラム中でのカットの使い方の一つは，手続き型のプログラミング言語の条件分岐と同じような働きをさせる場合です．たとえば，上の a という述語では，カット・オペレータを使うことによって，

```
if b1 then c1, d1
else if b2 then c2, d2
else if b3 then c3, d3
else if b4 then c4, d4
else e
```

に相当する手続きが実現されています．

また，否定の表現もカットの典型的な使い方です．たとえば，

```
not(P) :- P, !, fail.
not(_).
```

とすれば，引数となる項が成功したときに失敗し，失敗したときに成功する述語 not を作ることができます．

しかし，カットの使い方についてはここではこれ以上深入りせず，以降に示すプログラムの中で，具体的な使い方と働きを見てもらうことにします．

変数の利用

▶リストの要素を調べる

Prolog の重要なプログラミング技法のひとつに，「変数の利用」があります．変数を利用すると，リストの処理がたいへん柔軟になります．以下，簡単な例をあげて説明することにします．

図5を見てください．述語 makelist は，与えられたリストから，そのリストを構成する要素を重複なく取り出したリストを作るものです．

〔図5〕リストの構成要素を調べる

```
makelist([],L) :- !.
makelist([W|Res],L) :-
        member(W,L), !,
        makelist(Res,L).
member(X,[X|_]) :- !.
member(X,[_|L]) :- !,
        member(X,L).
```

makelist は再帰呼出しによって単純なループを作っています．その中では，与えられたリスト（第1引数）の各要素を順に member に送るという仕事をしています．第2引数は最初は変数です．これがどのように変化するかに注目して makelist の実行過程を追跡してみましょう．

いま，

?- makelist([you,can,not,can,can,in,a,can],L).

というゴールを入力するとします．最初は

member(you,L)

が呼ばれます．第2引数が変数というのは，これまでの member の意味からすると変ですが，そこに目をつぶると，これは member の第1節で，ただちに

L = [you|_]

という代入による単一化が行われることになります．

つぎに，ゴール節は

?- makelist([can,not,can,can,in,a,can],L).

となって，再帰的に makelist が呼ばれます．ただし，L は [you|_] になっていることに注意してください．これを，変数に名前を付けて，

L = [you|L1]

と書いておきましょう．このとき，

member(can,L)

を実行すると，

member(can, [you|L1])

が呼ばれることになり，再帰的に

member(can,L1)

が呼ばれて，

L1 = [can|_]

となって帰ることがわかります．すなわち，

L = [you,can|_]

が得られるわけです．

このように，ここでの member(X,L)は，「リスト L を先頭から調べて，X がすでに L に含まれていれば何もせず，まだ含まれていなければ L の末尾に追加する」という働きをしているのです．

この member の働きにより，makelist が再帰的に呼び出されるたびに，リスト L は

L = [you|_]
L = [you,can|_]
L = [you,can,not|_]
L = [you,can,not,in|_]

と後ろへのびていきます．そして

L = [you,can,not,in,a|_]

となったところで，makelist の第1節が成功し，プログラムは

yes

を返して止まります．

前回は，「Prologのリスト処理では，つねにリストの先頭を操作の対象とするのが基本である」と説明しました．しかし，じつはこのように変数を使うと，リストの後ろへ要素を追加することが容易にできることになります．任意の長さのリストの末尾の要素の操作が，変数の利用によって可能となるのです(ちなみに上の例文の意味は，「君は缶の中に缶を缶詰することはできない」です)．

▶ 2分木

さて，上で紹介したmemberを用いたプログラムは，あまり効率がよくありません．なぜなら，述語memberはリストを先頭から末尾まで毎回総当たりで調べるので，要素の数 n にたいして n^2 のオーダの時間がかかってしまうからです．そこで，効率よい処理が行えるように2分木を用いてmakelistを作り直したのが，**図6**のmakelist2です．

述語makelist2の二つの節は，基本的には**図5**のmakelistとまったく同じです．ただ，makelist2では述語memberのかわりに述語btreeが使われています．

述語btreeでは，2分木が作り出されます．ここでは，btを名前とする3引数の項によって2分木が表現されています．bt(N,L,R)のNは2分木の節点(node)，Lは節点より左側の部分木，Rは節点より右側の部分木を表しています．L，Rにさらにbtが入ることによって，いくらでも大きな2分木を表現することができます(**図7**)．

makelist2の第2節では，リストの要素を頭から順番にbtreeに引渡しています．btreeでは引き渡された要素について，すでに作られた2分木に同一の要素があるかどうかを調べます．このとき，要素と節点の大小関係にしたがって2分木を探していきます．

すなわち，要素の値が節点よりも小さい場合は左の部分木を探し，節点よりも大きければ右の部分木を探します．

この方法ですと，総当たりで調べるmemberよりもはるかに効率のよい探索が行えます．memberの比較回数が n^2 のオーダであったのに対して，こちらは $n\log n$ のオーダですみます．

探索の結果，すでに2分木中に同じ要素があれば何もせずに帰ります．まだ2分木になければ，ノードとの大小関係に応じて，2分木の末端の正しい位置に新しい要素を部分木の形で埋め込みます．

このようにしてmakelist2は，再帰的な処理によってリストが空になるまで2分木を作り続けます．たとえば，

?- makelist2([you,can,not,can,can,in,a,can],BT).

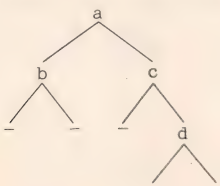
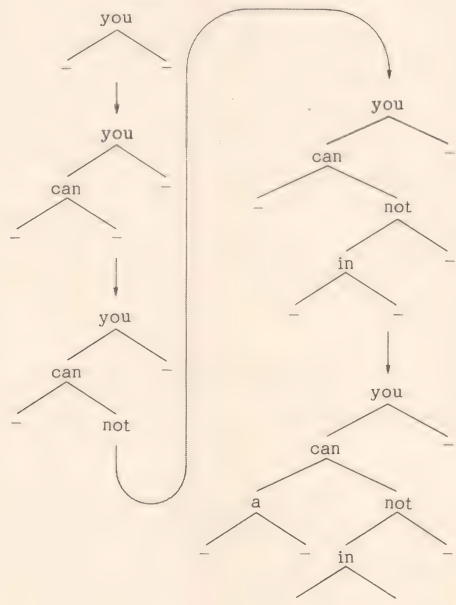
というゴールを入力すると，makelist2が再帰的に呼び出されるたびに，BTは

BT = bt(you,_,_)

BT = bt(you,bt(can,_,_),_)

〔図6〕 2分木を作る

```
makelist2([],BT) :- !.
makelist2([W|Res],BT) :-
        btree(W,BT), !,
        makelist2(Res,BT).

btree(K,bt(K,L,R)) :- !.
btree(K,bt(N,L,R)) :- K @< N, !,
        btree(K,L).
btree(K,bt(N,L,R)) :- K @> N, !,
        btree(K,R).
```

〔図7〕 bt(b,bt(a,_,_),bt(c_,bt(d,_,_)))の2分木

〔図8〕 2分木の成長

BT = bt(you,bt(can,_,bt(not,_,_)),_)
BT = bt(you,bt(can,_,bt(not,bt(in,_,_),_)),_)
BT = bt(you,bt(can,bt(a,_,_),
　　　　　bt(not,bt(in,_,_),_)),_)

とのびていきます．

図8は，この2分木が成長する様子を図示したものです．

差分リスト――キュー

▶差分リストの考え方

リストの操作をいっそう柔軟にする手法として，**差分リスト**(difference list，**dリスト**や**重リスト**ともいう)という考え方があります．これは，ポインタの対で一つのリストを表現する方法です．

普通のリストは末尾がnilで終わっています．これに対して，差分リストでは，それを変数のままにしておきます．前の節で見たように，こうしておくと，リストの末尾へのアクセスが可能になるのです．そこで，リストの先頭を指すポインタと，末尾から出ているポインタ(変数)とを一緒に管理しようというのが，差分リストの基本的な考え方です．

たとえば，リスト

X = [a,b,c|Y]

が与えられているとき，

X − Y

を，a，b，cを要素とする差分リストと考えます．ただし，このマイナス記号は，たんに二つの要素をつないでいるだけで，計算する働きはもっていないことに注意してください．X+Yでも，(X,Y)でも，sabun(X,Y)でもよいのですが,「差分」の気分を出したいためにマイナスの演算子を使っているにすぎません．

さて，この差分リストの末尾にもう一つの要素を追加して，新しい差分リストを作ってみましょう．そこで新しい変数Zを導入して，

Y = [d|Z]

と置いてみます．こうすると，

X = [a,b,c,d|Z]

になりますから，

X − Z

が，a，b，c，dを要素とする新しい差分リストになっていることがわかります．

〔図9〕キューの操作

```
enQ(X,L-[X|T],L-T).
deQ(X,[X|L]-T,L-T).
```

▶キューの操作

上のことを利用してキュー(queue)を操作するプログラムを作ってみましょう．キューとは，銀行の窓口などで見られる待ち行列と同じ構造で，最初に入った要素が最初に処理されるデータ構造のことです．このような構造は，スタックのLIFO(last-in first-out)に対して，**FIFO**(first-in first-out)と呼ばれています．

プログラムは**図9**のとおり，たった2行です．enQは一つの要素を差分リストの末尾に追加して，新しい差分リストを作るもので，deQは差分リストの先頭から一つの要素を取り出し，残りを新たな差分リストとするものです．

▶リストの結合

二つのリストを結合することも，差分リストを利用すると簡単にできます．

たとえば，二つの差分リストX−YとU−Vが

X = [a,b,c|Y]
U = [p,q|V]

のように与えられたとしましょう(**図10**)．ここで，Y=Uという単一化を実行すると，**図10**の点線のようなポインタが張られ，

X = [a,b,c,p,q|V]

というリストが生まれます．すなわち，

X − V

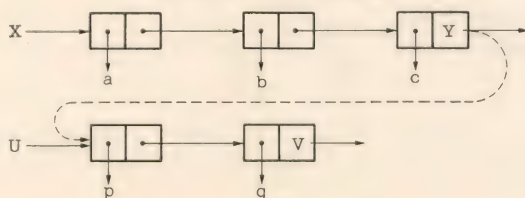
が結合の結果の差分リストになるわけです．

このようにリストの結合は一瞬でできてしまうのです．以前にappendという述語を紹介しましたが，それと比較すると，差分リストを利用した方法は，非常に優れているということができます．

▶簡単な構文解析

じつは，差分リストは自然言語の構文解析においてたいへん威力を発揮する手法です．自然言語処理の詳細については，この連載でいずれ取り上げる予定ですが，ここでは差分リストを使ったごく簡単な構文解析の例を見ておくことにします．

文をPrologのデータとして扱うために，とりあえず文を単語のリストとして表現することにします．たとえば，「瓢箪がたくさん実る」という文は，

〔図10〕差分リストの結合

[瓢箪, が, たくさん, 実る]

というリストで表します.

構文解析の基本的な操作は文の分解です. 上の文は

主部([瓢箪, が])

述部([たくさん, 実る])

のように分解することができます. しかし, このときいちいちリストそのものを操作して分解していては効率がよくありません. そこで, 差分リストを使い,

[瓢箪, が]

という部分を

[瓢箪, が, たくさん, 実る]−[たくさん, 実る]

というリストの対で表現すると, もとのリストはそのままで, 分解するのと同じ結果を得ることができます.

このような差分リストの利用は, ポインタを2本ずつ使ってリストの1部分を表現していると考えるとわかりやすいと思います. **図11** は, この考え方を図で示したものです.

差分リストを使うと, たとえば

<文>→<主部><述部>

という構文規則を

文(S0−S) :− 主部(S0−S1), 述部(S1−S).

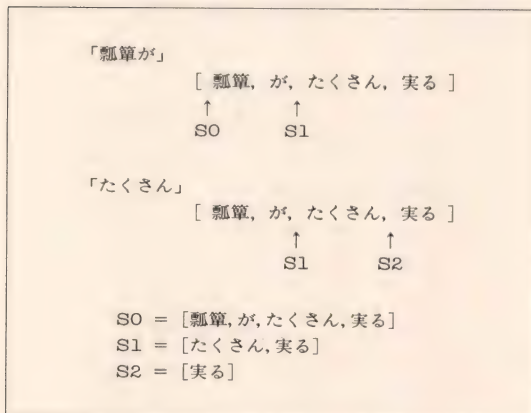
のように表現することができます. S0, S1, S はそれぞれ単語のリストですが, これをポインタと同等のものとみなしてもよいでしょう.

図12 を見てください. これは, 三つの名詞, 二つの動詞, 四つの格助詞を用意して, これらからなる文を構文解析するプログラムです. cp と sentence が構文規則を表現した述語です. cp は

<格助詞を伴う句>→<名詞><格助詞>

という構文規則を表し, sentence は

<文>→<格助詞を伴う句><動詞>

<文>→<格助詞を伴う句><文>

という構文規則を表しています.

たとえば, このプログラムに対して

?− sentence([瓢箪, から, 駒, が, 出る]−[ ]).

というゴールや,

?− S = [瓢箪, で, 鯰, を, 押さえる]−[ ]).

というゴールを与えれば, 構文解析に成功し,

yes

が返ってきます. また,

?− S =[瓢箪, から, 出る, 駒].

というゴールを与えれば, 構文解析に失敗するので

no

が返ってきます.

全解探索 —— パズル

Prolog を使うと, パズルのような全解探索の問題を解くプログラムがたいへん簡単に作れます. これは, Prolog がバックトラックという特有の仕組をもっているからです. Prolog はある意味でパズル向きのプログラミング言語ともいえます. 以下, おなじみのパズルで Prolog 言語の巧みさと面白さを味わってください.

▶金送れ

図13 は, 有名な覆面算です. アルファベットには0から1までの数字が一つずつ対応しています. **図14** は, これを解く Prolog のプログラムです.

まず ① では, 組込み述語 functor を使って, dtable というファンクタをもつ引数10個の述語を作っています. この述語は, 覆面算を解く過程で, すでに数字と対応させたアルファベットを登録していくテーブルの役目をします. 10 の引数は 0 から 9 までの数字と対

〔図11〕 差分リストの考え方

```
「瓢箪が」
        [ 瓢箪, が, たくさん, 実る ]
          ↑       ↑
          S0      S1

「たくさん」
        [ 瓢箪, が, たくさん, 実る ]
                  ↑         ↑
                  S1        S2

   S0 = [瓢箪, が, たくさん, 実る]
   S1 = [たくさん, 実る]
   S2 = [実る]
```

〔図12〕 日本語構文解析

```
noun(S0-S) :- S0=[瓢箪|S].
noun(S0-S) :- S0=[駒|S].
noun(S0-S) :- S0=[鯰|S].

verb(S0-S) :- S0=[出る|S].
verb(S0-S) :- S0=[押さえる|S].

case(S0-S) :- S0=[が|S].
case(S0-S) :- S0=[で|S].
case(S0-S) :- S0=[を|S].
case(S0-S) :- S0=[から|S].

cp(S0-S) :- noun(S0-S1),case(S1-S).

sentence(S0-S) :- cp(S0-S1),verb(S1-S).
sentence(S0-S) :- cp(S0-S1),sentence(S1-S).
```

応しています．

この dtable に対し，述語 enter(K,DTB,A)は，アルファベット A を数字 K と対応させて K+1 番目の引数として登録する述語です．もし，すでに他のアルファベットに対応している数字に重複して異なるアルファベットを登録しようとすれば，enter は失敗し，バックトラックが起こります．

プログラムは最初に 'M' を 1 と決めて dtable に登録しています(②)．これは，和の万の位を見れば明らかです．

③ は，'D' と 'E' の和の一の位が 'Y' であるという条件を調べています．そのために，まず 'D'，'E' を適当な数字に仮定して，DTB に登録しています．ただし，このとき 1 にはすでに 'M' が登録されているので，選ばれるのは 1 以外の数字です．もし，仮定した 'D' と

〔図13〕覆面算

```
      S E N D
  +   M O R E
  -----------
    M O N E Y
```

〔図14〕覆面算を解くプログラム

```
send([[S,E,N,D],[M,O,R,E],[M,O,N,E,Y]]) :-

        functor(DTB,dtable,10),                      …①

        M = 1,           enter(M,DTB,m),             …②

        digit(D),        enter(D,DTB,d),  ┐
        digit(E),        enter(E,DTB,e),  │
        S1 is D+E,                        ├…③
        C1 is S1/10,                      │
        Y is S1-C1*10,   enter(Y,DTB,y),  ┘

        digit(N),        enter(N,DTB,n),  ┐
        digit(R),        enter(R,DTB,r),  │
        S2 is N+R+C1,                     ├…④
        C2 is S2/10,                      │
        E is S2-C2*10,                    ┘

        digit(O),        enter(O,DTB,o),  ┐
        S3 is E+O+C2,                     ├…⑤
        C3 is S3/10,                      │
        N is S3-C3*10,                    ┘

        digit(S),        enter(S,DTB,s),  ┐
        S4 is S+M+C3,                     ├…⑥
        M is S4/10,                       │
        O is S4-M*10.                     ┘

enter(K,DTB,A) :- K1 is K+1, arg(K1,DTB,A).

digit(0).  digit(1).  digit(2).  digit(3).
digit(4).  digit(5).  digit(6).  digit(7).
digit(8).  digit(9).
```

'E' の数字の組合せが条件にあわなければ，バックトラックが起こり，条件にあう組合せが見つかるまで数字を選び直します．

④～⑥ は，③ と同様です．たとえば ④ は，'N' と 'R' に繰上げを加えた数の下位の数字が 'E' であるという条件を調べています．1 の位の計算によって生じる繰上げは，C1 によって ③ から引き渡されています．

以上のプログラムに対して

?− send(Answer).

というゴール節を与えると，しばらくして

Answer = [[9,5,6,7], [1,0,8,5], [1,0,6,5,2]] ;
no

という答えが返ってきます．

このように，Prolog で書くとプログラムはきわめて簡単明瞭です．手続き型の言語を使って同じプログラムを書くときの煩雑さを想像してみてください．

▶虫喰い算

図15 は，わずか 3 箇所の数字しか与えられていない除算の虫喰い算です．一見すると途方に暮れますが，Prolog ならば恐れることはありません．条件をひとつずつ書いていけばそれでプログラムは完成します．**図16** がそのプログラムです．

まず，道具立てとして，述語 add と述語 mul を用意します．説明のために，百，十，一の位の数字がそれぞれ a_1, a_2, a_3 である数を a_1-a_2-a_3 と表すことにすると，add(A1,A2,A3,B1,B2,B3,S1,S2,S3)は，A1-A2-A3 と B1-B2-B3 の和が S1-S2-S3 であることを調べる述語です．9 個の引数にはいずれも 0 から 9 の数字が入ることが想定されています．

mul(A1,A2,B,P1,P2,P3)は，A1-A2 と B の積が P1-P2-P3 であることを調べる述語です．やはり，引数には 1 桁の数字が想定されています．

sanketa(D,D1,D2,D3)は，3 桁の数 D の百，十，一の位の数字を，それぞれ D1，D2，D3 とする述語で

〔図15〕虫喰い算

```
              Q2  Q3   0  Q5  Q6
           ------------------------
 Y1  Y2  ) X1  X2  X3  X4   2  X6
           A1  A2
           --------
           B1  B2  B3
               C2  C3
           ----------------
                   D3  D4  D5
                   E3  E4  E5
                   -----------
                       F5  F6
                       F5  F6
                       -------
                            0
```

〔図16〕虫喰い算を解くプログラム

```
mushi([[X1,X2,X3,X4,X5,X6],[Y1,Y2]]) :-

    X5 = 2,                                          …①

    B3 = X3,
    D4 = X4,
    D5 = X5,
    F6 = X6,

    digit(Y1),                        Y1 > 0,   }…②
    digit(Y2),

    digit(Q6),                        Q6 > 0,   }…③
    mul(Y1,Y2,Q6, 0,F5,F6),

    digit(Q5),                        Q5 > 0,   }…④
    mul(Y1,Y2,Q5,E3,E4,E5),           E3 > 0,

    add( 0, 0,F5,E3,E4,E5,D3,D4,D5),                 …⑤

    digit(Q3),                        Q3 > 0,   }…⑥
    mul(Y1,Y2,Q3, 0,C2,C3),

    add( 0, 0,D3, 0,C2,C3,B1,B2,B3),  B1 > 0,        …⑦

    digit(Q2),                        Q2 > 0,   }…⑧
    mul(Y1,Y2,Q2, 0,A1,A2),

    add( 0,B1,B2, 0,A1,A2, 0,X1,X2).                 …⑨

add(A1,A2,A3,B1,B2,B3,S1,S2,S3) :-
    S is (100*A1+10*A2+A3)
       + (100*B1+10*B2+B3),
    sanketa(S,S1,S2,S3), !.

mul(A1,A2,B,P1,P2,P3) :-
    P is (10*A1+A2)*B,
    sanketa(P,P1,P2,P3), !.

sanketa(D,D1,D2,D3) :-
    D1 is D/100,
    T  is D - D1*100,
    D2 is T/10,
    D3 is T - D2*10, !.

digit(0).   digit(1).   digit(2).   digit(3).
digit(4).   digit(5).   digit(6).   digit(7).
digit(8).   digit(9).
```

す．

これだけの準備ができればあとは簡単です．まず，あらかじめ明らかな関係を記述しておきます(①)．つぎに，除数を適当に仮定します(②)．そして，あとは適当に数字を仮定しながら筆算式の下のほうから順番に条件を記述していけばよいのです．

③は，Y1-Y2とQ6の積がF5-F6である条件を記述しています．Q6が0ではないという条件も忘れてはいけません．④，⑥，⑧も同様に，除数と商のある位の数字との積の関係を記述しています．

⑤は，F5とE3-E4-E5の和が条件を記述しています．また，D3は0ではなく，D5，⑦，⑨も同じように，商を求める過程に現れる和の関係を記述しています．

以上のプログラムに対して

?− mushi(Answer).

というゴール節を与えると，しばらくして

Answer = [[5,9,1,5,2,8], [4,9]] ;

no

という答えが返ってきます．

▶エイト・クイーン

エイト・クイーンとは，8×8のチェス盤の上に，8個のクイーンを互いに取り合うことのないように置くパズルです．チェスのクイーンは将棋の飛車と角行を併せもったもので，縦横斜めどこまでも動けます．今回は，nクイーンにも対応できるプログラムを紹介します．

〔図17〕エイト・クイーンを解くプログラム

```
queen([],[],_).
queen(Index,[Q|Below],Above) :-
    member(Q,Index,Index1),
    check(1,Q,Above),
    queen(Index1,Below,[Q|Above]).

check(_,_,[]).
check(N,Q,[A|Above]) :-
    Q =¥= A+N,
    Q =¥= A-N,
    N1 is N+1, !,
    check(N1,Q,Above).

member(X,[X|L],L).
member(X,[A|L],[A|LL]) :- member(X,L,LL).
```

図17がそのプログラムです．これに対して，

?− queen([1,2,3,4,5,6,7,8],Config, [ ]).

を与えると，

Config = [1,5,8,6,3,7,2,4] ;

Config = [1,6,8,3,7,4,2,5] ;

Config = [1,7,4,6,8,2,5,3] ;

……

のような答えが返ってきます．最初の解は，「第1段の第1列，第2段の第5列，第3段の第8列，…にクイーンを置く」という解です(図18)．一般にnクイーンの場合には，第1引数として1からnまでの数の任意

〔図18〕 Config ＝ ［1,5,8,6,3,7,2,4］ に対応する解

Q							
				Q			
							Q
					Q		
		Q					
						Q	
	Q						
			Q				

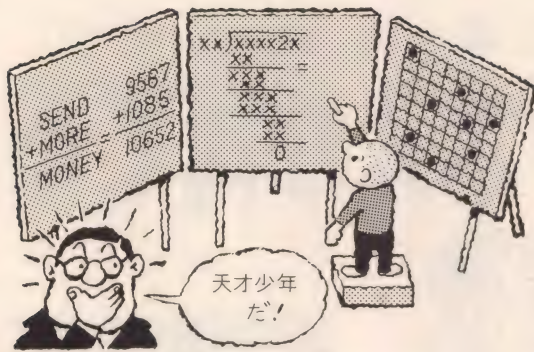
の順列を与えておきます．

まず，「クイーンが横に2個以上並ばない」という条件は，解がn個の数字の並びのみによって与えられるということですでに満たされています．

queen(Index,Config,L)は，与えられたIndexの数字の並びを並べかえて，エイト・クイーンの条件を満たす数字の並びConfigを返すプログラムです．まず，Indexで与えられた数字の並びから，何列目にクイーンを置くかを示す数字として一つずつ取り出します．そして上の段から順にクイーンを置いていき，逐次「自分より上のクイーンの利き筋にない」という条件を満たすかどうか調べます．

条件が満たされていれば，その数字がqueenの第3引数(Above)のリストの先頭に加えられ，つぎの段のクイーンの位置を調べるためにqueenが再帰的に呼び出されます．条件が満たされていなければバックト

〔図19〕 多項式微分のプログラム

```
diff(F,X,DF) :-
    diff1(F,X,DF1),
    reform(DF1,DF).

diff1(+F,X,DF) :- !,
    diff1(F,X,DF).
diff1(-F,X,-DF) :- !,
    diff1(F,X,DF).
diff1(F+G,X,DF+DG) :- !,
    diff1(F,X,DF),
    diff1(G,X,DG).
diff1(F-G,X,DF-DG) :- !,
    diff1(F,X,DF),
    diff1(G,X,DG).
diff1(F*G,X,F*DG+DF*G) :- !,
    diff1(F,X,DF),
    diff1(G,X,DG).
diff1(F^N,X,N*DF*F^(N-1)) :- !,
    diff1(F,X,DF).
diff1(X,X,1) :- !.
diff1(Y,X,0) :- !.

reform(-F,Ref) :- !,
    reform(0-F,Ref), !.
reform(F+G,Ref) :- !,
    reform(F,F1),
    reform(G,G1),
    ( ( integer(F1), integer(G1) ), !, Ref is F1
+G1
    ; G1 = 0, !, Ref = F1
    ; F1 = 0, !, Ref = G1
    ; Ref = F1+G1
    ), !.
reform(F-G,Ref) :- !,
    reform(F,F1),
    reform(G,G1),
    ( ( integer(F1), integer(G1) ), !, Ref is F1
-G1
    ; G1 = 0, !, Ref = F1
    ; F1 = 0, !, Ref = -G1
    ; Ref = F1-G1
    ), !.
reform(F*G,Ref) :- !,
    reform(F,F1),
    reform(G,G1),
    reform1(F1,1,C1,[],L1),
    reform1(G1,C1,CC,L1,LL),
    ( LL = [], !, Ref = CC
    ; CC = 0,  !, Ref = 0
    ; reform2(CC,LL,Ref)
    ), !.
reform(F^N,Ref) :- !,
    reform(F,F1),
    reform(N,N1),
    ( N1 = 0, !, Ref = 1
    ; N1 = 1, !, Ref = F1
    ; F1 = 0, !, Ref = 0
    ; F1 = 1, !, Ref = 1
    ; Ref = F1^N1
    ), !.
reform(F,F) :- !.

reform1(F*G,C,CC,L,LL) :- integer(F), !,
    reform1(G,C,C1,L,LL),
    CC is F*C1, !.
reform1(F*G,C,CC,L,LL) :- integer(G), !,
    reform1(F,C,C1,L,LL),
    CC is G*C1, !.
reform1(F*G,C,CC,L,LL) :- !,
    reform1(F,C,C1,L,L1),
    reform1(G,C1,CC,L1,LL), !.
reform1(F,C,CC,L,L) :- integer(F), !,
    CC is F*C, !.
reform1(F,C,C,L,[F|L]) :- !.

reform2(1,[F],F) :- !.
reform2(C,[],C) :- !.
reform2(C,[F|L],R*F) :- !,
    reform2(C,L,R).
```

Prolog はパズルの天才!?

ラックして，クイーンを置く列を更新するために Index から数字を選び直します．

Index から数字を選ぶのには，3引数の述語 member を使っています．member(X,L1,L2)は，リスト L1 から任意の要素 X を取り出し，L1 から X を取り除いたリスト L2 を作る役割をします．

このmember の働きで，いったん Index から選ばれた数字は重複しては選ばれません．したがって，はじめに Index に重複した数字を与えなければ，それだけで「クイーンが縦に2個以上並ばない」という条件が満たされる仕掛けになっています．

残るは「斜めに2個以上並ばない」という条件です．これを調べるのが check という述語です．Above を，いま調べているよりも上のクイーンの配置，すなわちすでに調べたクイーンの配置を表す数字のリストとすると，check(N,Q,Above)は，「いま調べている段の第Q列につぎのクイーンを置いたとき，Above の N 番目の要素が新しいクイーンの右斜め上または左斜め上にない」ことを調べます．

以上の仕掛けで，「互いに取り合わない」という条件を満たすクイーンの配置を網羅的に調べていくことができます．

多項式の微分のプログラム

最後に，Lisp との比較を示すために，数式微分のプログラムを紹介します．

〔図20〕多項式微分のプログラムの実行例

```
?-diff(x,x,DF).
DF      = 1
yes

?-diff(-2*x-a,x,DF).
DF      = -2
yes

?-diff(x^2,x,DF).
DF      = 2*x
yes

?-diff(-3*x^2+4*x,x,DF).
DF      = -6*x+4
yes

?-diff(-3*x^2+4*x^2-x,x,DF).
DF      = -6*x+8*x-1
yes

?-diff((2*x+3*y)*(x-y^2),x,DF).
DF      = 2*x+3*y+2*(x-y^2)
yes

?-diff((2*x+3*y)*(x-y^2),y,DF).
DF      = (2*x+3*y)*(-2*y)+3*(x-y^2)
yes
```

図19 を見てください．diff(F,X,DF)は，式 F を X について微分した式が DF であることを表現した述語です．プログラムは，おおきく二つに分かれています．前半の diff1 は微分の公式を表現している部分，後半の reform は得られた数式をきれいに整理する部分です．

図20 に実行例をあげておきます．仕掛けは Lisp とほぼ同じですから，例を見ながら各自プログラムの解読をしてみてください．

*　　　　*

今回で，Prolog の解説を終わります．次回は，AI にしばしば現れる探索問題を考える予定です．

たなか・ゆういち　㈶新世代コンピュータ技術開発機構
うみの・びん　東京大学大学院

連載インタビュー●ニッポン・テクノ風土論●コンピュータ・テクノロジー 日本的創造の可能性を追って［第7回］

パッケージ・ビジネスが育てる技術者の創造意欲 ——浮川和宣氏に聞く

下田博次
(インタビュアー)

これまでのソフトウェア技術者は，言われたことを忠実に実行することが中心で，個人的な創造力が発揮しにくく，仕事の達成感が得にくかった．しかしこれからはちがう．個性的な発想とテクニックでソフトウェア商品をつくり出していける時代になった．

ジャストシステムとその創業者浮川和宣氏の成功がその証明のような気がする．もうすこし具体的にいえば，一太郎とか花子といったパッケージ商品がベストセラーになったことで，日本でもソフトの商品開発に必要な技術者のオリジナリティに関心が高まってきたのではないかという気がするのだ．

この点を浮川氏はどう考えているのか．またソフト・パッケージ・ビジネスを発想するとき，彼がどんなことを重視しているのか．

浮川氏は，「ソフトウェア・ビジネスはコンセプトが大切」というが，それは何を意味しているのか．ソフト・パッケージ・ビジネスのパイオニアの発想と，全員ヒーロー，ヒロイン主義をめざすユニークな人材育成をはかるジャストシステムの秘密を浮川氏からあらためて聞いてみたい．

パッケージ開発の魅力

下田 博次
(しもだ ひろつぐ)
1942年生まれ．早稲田大学商学部卒．情報技術と人間の問題を中心に取材するフリーランスのジャーナリスト．近著の『リクルート新集団主義の研究』で，疑獄を生んだリクルートという企業文化を分析している．

—— 日本でベストセラー・パッケージが出ないといわれていたのが，ジャストシステムさんの出現でソフト商品化時代が来たような気がするんですが，いかがですか．

浮川　いや，それほどではないと思いますよ．大きな流れが定まっていたという感じですよね．とくにパソコンで著しいと思うんですが，パッケージ化の潮流が，時がたつにつれ強まっていますよね．それに乗ってきたということじゃないですかね．

—— でも，最初からその流れを読んでいらっしゃって，ターゲットを絞っていたんじゃないですか．

浮川　そうですね．はじめから，そういう確信はあったと思いますよ．一太郎をつくったときもそうですが，基本的に徳島でわれわれがやれることを考えると，パッケージをねらうしかないという気でいましたからね．

—— それはどういうことですか．受注型のソフト開発では限界があるということなんですか．

浮川　そうです．地方では，力を入れていくらいいソフトを作っても，それにみあった十分な対価が望めませんからね．しかし**パッケージは，たくさん使ってもらえば，マーケットはどんどん大きくなる**．いいものさえ作れば**2倍，3倍どころか100倍にもなってお客がついていく**．そこに創造のための工夫とかオリジナリティといった価値が生まれる余地ができ，また努力することもできる．

—— かけ算の可能性がでてきて，技術者としてのやる気が大いに刺激されるというわけですね．しかし，聞くところによると，最初に

会社をおつくりになったとき，たいへん多額の受託開発ソフトの契約話が出てきて，迷ったということじゃありませんか．

浮川　それも事実なんです．当時の金額で6,000万円ぐらいの仕事がきたんですよね．それで迷ったこともあった．徳島で，それをやるのがいいのかどうか．最初からパッケージをねらっていたんですが，すぐにビジネスになるものでもない．それで心が動いたのですが，しかしそれをやると，私も家内(ジャストシステム専務)も縛られることになる．そうすると徳島で会社を起こした意味がない．だから，色々考えてみても，やっぱりこちらのほうが正解だということで選択をしたんですね．

―― 東京から来た大きな仕事の請負に縛られるのはいいことじゃないと判断されたわけですね．

浮川　そうです．(金額の大きさに)心は動いたのですが，もともとパッケージという考えが強くあったので，みきわめがついたということですね．

ジャストシステムの10年(同社会社概要から)

1979. 7.	ジャストシステム創立．オフコン・システム販売ディーラとしてスタート
1981. 6.	株式会社ジャストシステム設立
1982.10.	酪農システムのソフトを開発・販売 8ビットCP/M用の日本語処理システムをデータショウで発表
1983. 3. 10.	ワープロ・ソフトの開発を始める NEC PC-100用に標準採用されたワープロ・ソフト「JS-WORD」開発
1984. 5. 12.	日本語FEP文節かな漢字変換システム「KTIS2」開発 IBMパソコンJXシリーズ用ワープロ・ソフト「jX-WORD」発売
1985. 2. 8.	NECパソコンPC-9801シリーズ用ワープロ・ソフト「jX-WORD太郎」発売 「jX-WORD太郎」の後継として「一太郎」発売 東京サポート・センタ開設
1986. 5. 12.	「一太郎Ver.2」発売 NECラップトップ・パソコンPC-98LT用ワープロ・ソフト「サスケLT」発売
1987. 1. 3. 6. 12.	新居浜事業所開設 図形プロセッサ・ソフト「花子」発売 「一太郎Ver.3」発売 現在の本社ビル完成・移転
1988. 5. 6. 6. 12.	イメージ・プロセッサ「シルエット」発売 大阪サポート・センタ開設 日英ワープロ「duet」発売 イメージ・スキャナ「JS-SC201」発売
1989. 2. 4. 6. 6. 7. 8.	EMS対応メモリ・ボード「JS-EM201」発売 「一太郎Ver.4」発売 東京サポート・センタ/営業所，浜松町移転に伴い東京支社となる 本社第2ビル完成・移転 「花子Ver.2」発売 岡山研究所開設

次の次のパッケージを夢みる

―― ロジック・システムズ・インターナショナル社からの仕事が，そのパッケージ・ビジネスの手がかりになったということなんですが，そのへんをもう少し詳しく話してください．

浮川　ロジック・システムズ・インターナショナルさんとの契約はCP/MとMS-DOSのカナ漢字変換だけだったんですが，その後，私たちとしては，これだけじゃ世の中にモノは出せない．それであれこれ考えて，結局ワープロをやろうということにしたんです．それも日本語処理のよくできるものを．それが一番いいものであれば，日本語ワープロもいいものができるはずだと．

今から思うと，自分たちのワープロを作ろうということだったんですよね．当時のワープロは専用のイメージが強かった．しかし私たちとしては，最初からカナ漢字変換の部分をOSでもっていて，ワープロで辞書を自分流に改良して，それをデータベース化して，なんにでも使えるようなもの．つまり汎用コンピュータをめざしていたんですね．そうしなければ，もったいないんじゃないかという気持があったわけです．

―― そういう商品化コンセプトは，奥様でもあり専務さんでもある初子さんとの共同作業だったということですか．

浮川　そうです．こういうコンセプトは，私と家内と福良(同社・福良伴昭取締役)の3人でやってきたんです．でもそれは最初の頃のことで，最近ではもう3人でワイワイやっていられない状態になってきているのも事実です．

―― 奥様との二人三

浮川 和宣
(うきがわ かずのり)
1949年，愛媛県新居浜生まれ．1973年，愛媛大学電気工学科卒業後，船舶関係の西芝入社．6年後の1979年，ジャストシステムを設立．専務の浮川初子夫人とは，大学時代のアマチュア無線部で知りあったという．大学の専攻に電気を選んだのは，「電気が一番難関そうにみえたから」で，無線部に入ったのも「専門に行く前に無線を勉強しておこうと思ったから」

「最初にコンセプトありき」でコンセプトが人(エンジニア)を要求してるんです

脚の話は後ほどまたおうかがいするとして，その前に一太郎などパッケージ商品の最新の実績について説明していただけませんか．

浮川　**一太郎は現在バージョン 4.2 ですが，トータルで 45 万本という実績**になりました．ただバージョン 4.1 でバグがいろいろ指摘されまして，その対応を終えたところです．現在のバージョン 4.2 ですが，これは自信作です．

―― 初期のバグは深刻だったんですか．原因はどんなところにあったのでしょう．

浮川　ユーザ・サイドではお困りだったと思いますし，私たちとしてもとらえきれなくてご迷惑をおかけしたわけですが，原因は，結局，ボリュームに対するテストの時間が不足していたことにあったのだと思っています．**バージョン 3 のころは社員が 100 人でしたが，今は 330 人という規模**になっていますから，それと，パッケージがコンシューマ製品として認識されるようになってきて，バグに対する見方もすさまじく変わってきましたからね．

―― ユーザのすそ野の広がりと，一方ではイノベーションの急激な展開があるわけですが，32 ビット時代の戦略はどういうことになるわけですか．

浮川　私どもは，AAC というコンセプトを発表しています．パッケージは NEC さんとか IBM さんとか別々に出しますが，品質管理ということでは枯れたモジュールをどんどん活用していこうという作戦です．もともと，私としては，次の次のバージョンを夢みて，ロング・スパンでやってきましたから，AAC というコンセプトもそのひとつですね．

―― ユーザにとって理想的な利用環境を先へ先へと追い続けていくということでしょうか．

浮川　そうです．色々なソフトとの関連をもっとやりやすくするとか，ユーザの手になじんだものにできるには，どういう広がりが必要かとか，そんなことを長いスパンで考えるんですよ．

―― 最初の一太郎のときも，先を考えていらしたわけですね．

浮川　そうです．一太郎についていえば，バージョン 4 で，最初のころ考えていた夢が実現したということでしょうかね．

いいコンセプトが宝の山に

―― さきほど奥様である専務さんとの二人三脚の話がでましたが，そういう商品化の夢は 2 人というか，3 人でおやりになってきたわけですね．

浮川　まあ，どちらかというと，私は好きなことをいいたい放題でやってきて，大まかな方向をつけてきましたが，家内のほうは(商品化の夢を)現実のものにする力がありますからね．私は，「なんでそんなことすぐにできないのか」といいますと，「そんなに早くは実現しませんよ」といいかえされたりするわけです．

―― 家に帰っても，2 人でディスカッションをしているということですが，息が抜けないということはありませんか．

浮川　私たちはいつまでこんな生活をするのかしら，と笑っていうことはありますね．私はどちらかといえば営業の立場で，家内は生産のほうですから，かなり対立点はありますよ．しかし，結果的に，こんな風なものができるといいな，といっていた夢がつぎつぎに実現してきたわけですからストレスはありませんよ．それに最近では，2 人していたわりあうような場面が多くなりましたね(笑い)．

―― その夢の実現の条件なんですが，コンセ

(ジャストの)平均年齢は約26歳
力を発揮するのは入社3年目ぐらいからですね

プトがいいものでもそれぞれをどう実現するか，一番の問題はどこにあるのでしょうか．

浮川 人のパワーでしょうね．**100人のパワーが必要なコンセプトを出したときに20人じゃしょうがないですからね．**まあ，人がふえればマネージメント・パワーもそれだけ必要になるわけですが，仮に500人いればここまでできるとわかったとき，それをなんとかしなくてはいけないわけですね．**マンパワーの大きさで，ソフトウェアの質も変わってくる．**とくにトータル環境を良くしようとすればするほど人が必要になってくるわけです．

―― マンパワーとソフトの中味は比例しているわけですか．しかし，人がたくさんいれば良いものができるというわけでもないでしょう．前提としてコンセプトの中味，質がよくないと，いくら人をつぎこんでも意味がない．むしろいいコンセプトが，いい人とマンパワーを必要とするということじゃないでしょうか．

浮川 おっしゃるとおりだと思います．これは，もう私の信念なんですが，いいコンセプトがあってはじめて製品の方向，ひとつひとつの機能が生きてくる．いいコンセプトはソフトウェア・ビジネスにとっては宝の山のようなものなんですね．

―― 結局，浮川さんのお仕事というのは，長期的にエンド・ユーザの立場に立ってパソコンのありかたをとらえることと，そのビュー・ポイントから個々に実現される機能と製品系がいつも有機的につながっていくよう統合化したイメージを描きつづけるということでしょうか．

浮川 そうですね．そういわれてみて，あらためて，本来の私の仕事はそういうことだったと思うわけですが，最近は経営とか組織とかの仕事が多くなってきまして，そのへんのことになかなか力を注げなくなっているのも事実なんです．

―― 経営の仕事でたいへんということは，財務などもみているわけですか．

浮川 いや財務のことは，会長がしっかりやってくれていますので安心ですが，人事ですね．この仕事が多くなってきた．

適材適所をいかに実現するか，人の問題が最後は重要になりますよね．研究所をつくるとか，未来にむけての楽しい仕事もありますが，それにしても現実に立ち上げていくためには人の問題が避けてとおれないわけです．

気になる自己規制と気配り

―― 人の問題といいますと，適材適所もさることながら，どうしたらいい人材を採用できるか，ということも大きな問題ですよね．その採用の点で，ジャストシステムが徳島にあるということがメリットになっているのでしょうか．聞くところによると，お宅の会社では新入社員に2 LDKとかのマンションを与えていらっしゃるということですが．

浮川 2 LDKでなく，2 DKなんですが，まあ東京でいえばマンションくらいのものでしょうね．東京の住宅事情が悪すぎるんですよ．ですから，その点で徳島は恵まれていますよね．また情報の格差というものもありませんしね．やはり徳島でいいと思いますよ．

―― これまでジャストシステムは，浮川さんのソフト・コンセプトを実現する組織として急速に大きくなってきたわけですが，これからは，浮川さんのその発想を超えるというか，それとは次元の違った異質な発想ができる人を育てたり，集めたりしていかなくてはいけない．そんなお考えはありますか．

浮川 そこのところを，今いっしょうけんめい考えているところなんです．もとから，そういう考えを述べてきたのですが，それは一貫して変わらないのです．

私としては，最近うれしいことがありまして，彼がこんなことを考えていたの，というような提案が，先週，1週間の間に2件ありまして，家内とも評価しているんです．ちょうど，新しくつぎのプロジェクトをスタートさせなくてはいけない時期なので，うれしい動きですよね．

こちらが思っていなかったようないい提案が出てくると，みんなそれぞれ夢がある人だなぁ――とうれしくなりますよね．こういう

**日本国内のパソコンは約500万台
この500万台をわずか数百人のエンジニアがささえている
そう思うと大変なやりがいですね**

と，少しきれいごとに聞こえるかもしれませんが，自分が思っている以上に若い人が育っているな，と実感したときは素直にうれしいですね．

── そういう若い人たちの提案が実際にプロジェクトとして動き出し，それがマーケットで成功を収めていくと，ジャストシステムの社内の雰囲気も変わっていくでしょうね．

浮川　そうです．現におもしろいことをいっているな，こいつ，ひとつやらせてみようか，と思うことが増えていますね．いや，もう少しせっぱつまっていえば，こいつはやらせてみないとおさまりがつかなくなっているな，そう思うことがありますよ．

私としては製品開発などのエンジニアだけでなく全員がそういう方向にいって欲しいんですよ．

── これまでのヒット商品に代わる，新しい発想のパッケージがお宅の組織の中から出てくるといいですよね．

ところで，この対談では技術者の自己主張ということが共通のテーマになりそうなのです．新商品開発にも自己主張が必要で，そういう人をどう育てていくかが関心の的になっているわけですが，ジャストシステムではいかがですか．

経営者によっては，今の若い人で，オレはこうしたい，という主張をきちんといえる人間が少なくなっていると，なげく声もあるのですが．

浮川　日本の社会は，自己主張に慣れていないと思いますね．それをやると，はたに迷惑をかけるんじゃないかという，気配りのようなものが強くなっているというか，そういうふうに(いまの若い人が)育ってきているような気がしますね．

私のみるところでは，本当は自分の主張，やりたいことが強くあるのに，それがうまく表現できないとか，そうしたモヤモヤした状況があるように思うんです．そういう若い人のほうが多いというか．だから私としては彼らの気持を聞いてやることを心がけねばならないわけです．

色々言いたいことがあったら言ってもいいんだよ，とか，できるだけ誘い水をかけたりするんです．

── ほーう．そうすると，こんなことはいえませんか．つまり，今の若い人は受験戦争の中で学力とか能力というものは身につけているが，それとは別に夢を抱いて，それを実現するように動いていく，能動的な能力は弱くなっている．

浮川　そうですね．私のところでは，ともかく**皆がヒーロー，ヒロインになれるようにしようと努力しているんです．**それで評価のモノサシもたくさん用意しているんですが….

── そうはいかない状況もあるわけですか．

浮川　はっきりいって，まだまだ自己規制してしまう空気が強いですね．社長の僕が遠慮するな，といっているんだからどんどんやれよ，といっても，気配りのほうが出てしまうような状態ですね．

── ひょっとすると，自分がやりたいことを提案すると，逆に負担が重くなる．責任を引き受けたくないという安全心理が若い世代に出てきているかもしれませんね．とするとこれはジャストシステムだけの問題ではないかもしれない．

真の技術者として生きるため シリコンバレーで会社設立

八木 広満

自分はなぜIC設計を15年もやってきたのかわからなくなることが最近あった――

希望のIC設計部門に就職

小学生の頃から，電子部品を見ると何か胸がワクワクした．きっと目立ちたがり屋だったのだろう．人のわからないことをやっているのだぞという，子供仲間のヒロイズムでこの道に入ったような気もする．

何をやってもブレーキの効かない性格も原因しているに違いない．学校を選ぶにも電気屋になるための学科で入れるところを探したし，就職するにもMOS-ICの設計ができるところという一点で選んだ．会社では何かしら，「自分はたまたま，この部署に配属されて，たまたまその仕事をしているのではないのだぞ」という，いまから思えばずいぶんはた迷惑な気負いをもっていた新人エンジニアであったような気がする．

その会社(1974年4月入社)では，自分の希望どおりMOS-ICの産業用グループと称する設計部署に配属され1～2年は毎日が楽しかった．何でも一人でできるのだという若者特有のずうずうしさと，何も仕事らしい仕事をやらせてもらえないという不平が，じつに得手勝手な，もしそいつがいま自分の目の前にいれば張り倒しているに違いないような，人間を作っていた．――これは一人自分の姿ではない．すこしも言い訳をするつもりではないが，若い，元気があるということは，そんなことなのだろうと思う．さすがに，好きで入って来ただけに，少しは飲み込みが早かったのか，または人一倍のずうずうしさが効いたのか，人並に2～3年で仕事を理解することができるようになり，ついでに，職場結婚というものもした．入社時は24歳であったから，そのときは27になっていたように思う．

そのころの日本のIC設計は，米国のデッド・コピーといってもよいようなものであったと思う．会社も半導体事業そのものに力を入れようという気もなく，カンバン程度にもっているところがほとんどであったろう．もっとも，米国が技術的にも圧倒的に進んでいたし，その技術成長速度も驚倒させられるものであった．たしかに日本は，技術成長のリーダシップを取るのに向いていない組織構造だと思う．近年，日本の技術力が，米国を圧倒し，これを追い越したと日米双方が信じている向きがあるが，これはたんに米国がその成長を緩めたにすぎないと言い換えるべきではないか．

顕微鏡をとおして見えたもの

いずれにしても，その頃は何かオリジナルなICを設計させてもらいたいと思っても，そのチャンスもなく，また何を作ってよいのかわからず，現実に作る力もないという，三拍子が揃っていた．そして米国，とくにIntel社の新製品が出ると急いでこれを入手し，フタをこじ開け，顕微鏡で中をのぞき込むということをくり返していたように思う．風のたよりで，このICを設計したのは自分とそれほど年齢の違わないエンジニアで，いまではさらに進んだICの設計をほぼ終えかけている…ということを聞くと,「こんなことをしていてよいのか」「これが自分のベストなのか」と，なんともやりきれない暗い気持になったりした．

しかし，この鼻柱ばかり強い高慢な自分にも，その顕微鏡の下にあるICが，わずか6ヵ月以前のものに比べて自分の想像臨界点を越えて急激な仰角をもつ，進んだ技術のもとに作られたものであることを認める余裕は残っていた．「すごい!!」と思った．老成した人間にいわせれば，たんなるオッチョコチョイであるに違いないが，自分はいまではそのようにすなおに驚くことができる人間であることがエンジニアにとって不可欠の資質だと信じるにいたっている．日本的な2μmを1μmといった細かい改良で進めていくのではない．日本の改良はおおむね第三者から見ても想像のつくものの場合が多いが，当時の米国の改良(以後も)は，思いも寄らぬ工夫でブレークスルーを実現したものが多かった．「へぇーこんな方法があるのか…」である．

4～5年もこんな日々が続くと，いわゆる中堅エンジニアになり，社内の事情がわかり，性格も少し角がとれて会社では使える人間(戦力)になってくる．ところで終身雇用が原則の日本の大企業では，わずかな個人的能力を活かすよりは，全体の調和を乱さないほうがよいという場合が多い．その能力を活かすために，

周囲を除くことはできないし，それをやると，誰もが安心して終身雇用を信じず，会社への忠誠心が減少するに違いない．日本では誰もが自分の会社と信じている．だから，給与だけで信じられないほどの忠誠心を持続し，会社の利益に貢献しつづけるのである．このような環境にあって，個人の能力は(とくに技術的能力は)少々高いよりは，少々低いほうがまだ無害である．日本では教育そのものが画一的Bクラスの人材を生産するシステムになっており，この会社の形態とマッチして今日の大成功を見たのではないか？

シリコンバレーを見たい

自分は，この仕事が好きでこの道に入ったし，貧弱とはいっても，自分の技術能力に見切りをつけることはどうしてもできなかったので，少しでもオリジナルICの設計チャンスのある場所に移りたかった．ずいぶんと悩んだ．わがままである．Intel社のゴミ掃除でもよいから何とかもぐり込めないかと真剣に思ったりした．

曲折のあと，1979年7月に以前の上司の世話である事務機会社の新しくできる半導体部門へ移ることになった．その会社の殺し文句は，「アメリカの半導体屋を見せてやる」であった．いまでは信じられないことだが，当時の日本の大企業は，社員が外界を見ることを好まない部分があったのか，他社の内部を一介の取柄のないエンジニアが見るチャンスを得ることなど夢でしかなかった．この話には即とびついてしまった．いまさらながら，この上司には感謝にたえないのだが，この口約束を実行してくれた．その齢になって，約束とはどういうものかを教えられた気がした．

何度かのアメリカ出張(じつは仕事などない)で，自分はようやく夢のシリコンバレーを見るチャンスを得た．もちろん言葉はわからないし，仕事があって行っているわけではないので，半導体会社へ行くと，どんな道具を使って，どんな場所で，どんな人間が，どんなことをしているのか，またエンジニアはどんな扱いを受けているのか，そんなことばかりを尋ねていたように思う．Intelのマイクロプロセッサ部門に行ったとき，会議室に行くまでの通路に，8086のペン・プロット図が張ってあるのをチラッと見た．その一辺2m近い紙は，全体をいくつかの格子状に分割して作図し，それを寄せあつめたような線跡が，くっきりついていた．ああやっぱり全体をしっかり計画して，そのあとそれを分割し手分けして，あんなに芸術的につまったレイアウトをしているのだ，と感激した．何も魔術師がやっているのではない，それなりの方法を疑わずに実現していっているだけなのだと，何かホッとした．

1974年，大阪大学卒業後，大手電機メーカに入社．1979年，大手事務機器メーカに移り，1980年1月に最初の渡米．1984年，日本に戻らない覚悟で渡米，米AMI入社．その後，ベンチャCylink Corp.(1986年)などを経て，1988年にClarkspur Designをサンノゼで設立．

ICをつくる！　そして失望

その翌年に，その会社の技術レベルから見ると場違いともいうべきDSPチップを設計することになった．しかし曲がりなりにも本格的に作業が始まって6ヵ月程度でレイアウトが終わり，ヨタヨタながらサポート・ツールと呼べそうなものができそうになった．しかし，やはり，やっつけ仕事であり，その程度のリソースしか与えられていなかったこともあるのだが，出来上りは今ひとつだったように思う．また，“DSPって何？”という時代だったこともあり，唯一の社内顧客が，その応用に必要なスピードが得られなかったこともあり不採用を決めるとともに，プロジェクトは，自然消滅した．あとには，ユーザのないICだけが残った．商品だけあっても，その使い方，価値を知る力のない会社には，それを売ることはできない好例である．

結局，その会社は，その前後，自分も関係したオモチャ用のICを作っている．このオモチャ用のICの不良があるユーザで発生し，その後始末に出かけたことがある．昼間は工場が稼働しているので未実装基板に付いている不良ICを良品に交換する作業をするが，夕刻から工場が停止すると最終製品を箱から出して不良ICを見つけて交換するといった作業である．本社などからの応援20名ばかりを得て，約1週間かかった．応援の誰もがどういうわけか私を責めていた．そこには仲間という気持ちも何もなく，自分の人徳のなさを痛感し，同時に自分はいったい何をしているのだとも思った．少しでも良いエンジニアになるべく願い努力してきたつもりが，東北の冬の片田舎で，深夜にテレビを担いでいる．会社も，自分をその程度が似合いだといっているようで，情けなかった．

その一件以来，自分はエンジニアをやりたければ，米国へ行く以外にないと思い始めていた．

シリコンバレーの技術者になる

このオモチャICがなんとか量産になってしばらくして，自分はこの会社を辞め米国に行くことにした．最初の会社を辞めるときのような寂しさはなく，逆に，ふみはずした道をこれから元の正しい道にもどるのだという明るい前途を見ていた．しかし，もう34歳になっている．まず，DSPチップをやっていたときに関係のあった米国のAMI社の(2～3度会ったきりの)知

人に，エンジニアの空はないかという手紙を出してみた．ひどい英語だったと思う．好運にも当時米国は好景気の余波が残っており，自分でも不思議なくらいアッサリと採用されてしまった．女房と3人の子供はとりあえず日本に残したまま，相手の気の変わらないうちにと急いで米国へ行くことにした．飛行機への通路から窓越しにターミナル・ビルを見ると，思いも寄らず後輩2人が，手を振り回しているのが見えた．もうここから先は自分を知っている人間はいないのだなあと，はじめて心細くなった．1984年7月のことである．

AMI社に初出社した日，一日かけて他の同日入社の連中と一緒に入社手続きがあった．その中で，経歴では自分が最右翼であったにもかかわらず，その説明がほとんど一言半句わからなかった．現地人同士の大人の会話だからとはいわず，要するに自分は英語がわからないのだと思い知らされた．客で来ているのではない．金をとって働くことになっているのだ．その日から数週間は自分の気持を取り静めるのが精いっぱいであった．

しかし，なんとおもしろくて陽気な連中であろうか．日本風の指揮系統も，上司は(形だけでも)尊敬されるべきであるという考えも，ほとんど感じられない．エンジニアは，良い物を設計することが職務で，それができる者が尊敬を受ける．他のことは要求されない．数ヵ月自分は日本人的であった．少なくとも，その会社のエンジニアに比べれば，日本のどの会社の中堅エンジニアも，万能選手であり，歩く百科辞典と呼ばれてもおかしくないほどの知識をもっていたと思う．しかし，百科辞典である．つまり，何でも知っているが，物を作るのに必要な深い知識と能力がない．それらを吸収しようにも，こっちはオシでツンボときている．

ベンチャ企業への参加

結局自分はAMIに1年半いた．その間半導体不況があり，2回のレイオフを経験した．米国の会社は，経営が悪化すると，従業員を解雇するのである．もっとも従業員のほうも，景気が良くなると，良い条件の働き口を探してキョロキョロしだす．信頼関係どころか，会社と従業員はキツネとタヌキだ．

とはいいながらも，そこはシリコンバレーである．ときとして会う人々は優れた専門知識をもっていたり，ユニークな物を作っていたりする．そんな人や会社に車で15分でおおよそ行き着ける地域である．日本でも九州をシリコン・アイランドといったりすることがあるが，それは本当のシリコンバレーの意味を知らない人のいうことだと思う．半導体にまつわるあらゆる応用技術の専門知識と，金と，安っぽくいえば夢が渦をまいてうなりを上げている場所である．幸運にも自分は，AMI退社後，身をもってこれを知る機会を得た．

AMIでの1年半の間に自分は永久ビザを入手した．1986年2月，あるベンチャ・ビジネス会社でRSA暗号用のICなどを開発する誘いを受け，大きなストック・オプション(事業成功時の自社株購売権)とともに入社することになった．わずか5名のエンジニアで，自分の担当はICの設計である．学生時代に有名な学者として知っていた人が社長，CEOであった．自分以外のエンジニアは，誰をとっても，すばらしい専門知識と実行能力，問題解決能力をもっていた．いるところにはいるのだと思った．自分は言葉も不自由であり，そんな連中とやってゆくには，汗で対抗する以外にないと考え，とにかく働いた．しかしベンチャ・ビジネスではアメリカ人でもよく働くのである．ストック・オプションのために全員が金の元に結束しているというだけではない．これができるとすごいじゃないかという夢に，金が乗っかっているのだ．金だけでは人間そんなに働けるものではない．日本企業の駐在員が物知り顔に金の結末だけをいうのを見ると，口を開ける気もしなくなる．自分はすでに日本人でなくなりつつあるのかもしれない．

技術者としての再出発

この時期は，自分にとってももっとも幸せな時期であったと思うが，その会社で2品種のICを作ってしまうと，もうIC屋は不要になってしまった．社長以下，何もしなくてもよいからいろとまでいってくれたが，自分は，つぎのアテもないまま，辞めるべきだと思った．しかし，その会社には，いまでもよく呼ばれて行く．すでに，ずいぶん大きな会社になっているが，仲間は変わっていない．うまくいっている会社ではアメリカでも人の入れ替わりはないのだ．

その後，つまらない時期を日系の会社で駐在員として過ごしたが，1988年4月に思いきって，ICデザイン・ハウスClarkspar Designをシリコンバレーのサンノゼに開いた．そこで自分の作るべきだと思うICを自分の金で作るというのが考えであった．もちろん，生きるためにときどきコンサルティングもやるが，時間を切り売りしないで，責任を売ることにしている．最初に手がけたものは，モデム用DSPコアで，信じがたいことだが6ヵ月程度で設計が上がってしまった．日本の会社にいるときに比べると1/5のリソースしか使っていないことになる．シリコンバレーでは，個人が，ICを1から10まで作り通せるということは，ここに来て，やってみせないと信じてもらえない．

今年5月に，自前の最初のICがシリコンになって上がってきた．不思議と何の感激もなく，39歳という年齢を寂しく思っただけである．

やぎ・ひろみつ　Clarkspur Design

辛口テクノロジ批評

ソフトウェア製造現場の教育性を高めよう

Robert T. Myers

ソフト技術者は本当に足りないか

ソフトウェア・エンジニアが足りない．これはメーカ，ソフト・ハウス，民間企業，政府の共通の悲鳴になってきている．来たるべき情報化社会のインフラストラクチャを作りあげる人間がいなければ，どうしようもない．日本のメーカが相次いで海外で研究所を作っているのも，半分は向こうの顧客のニーズをより的確に把握するためであるが，残り半分はやはり日本国内の技術者不足に起因するだろう．

しかし，厳密にいえば，それはソフトウェア・マンの不足の問題ではない．むしろ，ソフトウェア人口かける一人当たりの生産性のかけ算，すなわちソフトウェアの総生産量が問題なのである．ここでは，この方程式の右側，すなわち生産性，に焦点を合わせよう．

ソフトウェア生産性の重要性

この生産性は，たとえば自動車産業のそれとはだいぶ違う．車だと，おそらく組立てラインでは新入社員と，超ベテランとでは3～4倍の生産性の差しかない(筆者は車産業の評論家ではないので断定はできないが)．また，工業の世界では人の生産性は比較的計りやすいと思う．教育とか訓練の時間数で，どれくらい上がるかも検討がつく．無事に動く車さえ作っておけば，5年後にたいへんな問題が突然現れてくることもあまりないだろう．班長が皆の仕事のでき具合を見て判断し，即決で問題の善処，人の再配置などを図れる．

しかし，ソフトの世界では，二次産業である鉄や車産業より，またソフトと同じ三次産業でも運輸などより，生産性の計測も，その向上策も，たいへん難しい問題である．またトップクラスの人と初心者との間の生産性の差は，一桁以上だと思われる．通産省などから発表される『1992年にはソフトウェア・エンジニアの不足が○○万人に昇る』というような調査結果があるらしいが，もしソフト生産性を倍にすることができれば，逆にプログラマがあまるくらいになるかもしれない．

生産性を考えるための例題と解答例

まず計測から考えよう．つぎの例題を見てほしい．

問：文字列を逆さまにする

これを，A，B，C，Dの四君に答えてもらった．

(1) A君の解答

プログラマのA君は図1のプログラムを組んだ．これに1時間かかったとしよう．評価はあとでまとめて行う．

(2) B君の解答

B君は図2のような，その場置換えにした．B君はちょっとしたテスト環境も作ったりして，徹底的にテスった*1ので，2時間かかったとしよう．

(3) C君の解答

C君は似たようにプログラんだ*2．ただ，置換えのところにtempを使わないで，下のような非常に変わった置換えをしている．

```
a[j] ^= a[i] ;
a[i] ^= a[j] ;
a[j] ^= a[i] ;
```

すなわち，aとbとを置き換えるために，3回の排他論理和を使うわけである．たとえば，A=0x0101でB=0x0011だとすると，

```
A ^= B    A=0x0110  となり
B ^= A    B=0x0101  となり
A ^= B    A=0x0011  となり
```

きれいに中間変数なしで置き換えができた．こういった手法をご存じでしたでしょうか．この技法の評価も後に回す．

(4) D君の解答

最後にはD君の登場である．彼の作品を図3にあげる．D君も，このプロジェクトに1時間をついやしたとしよう．

さて，どれに軍配はあがるだろう．

解答に対する評価

A君は失格である．バッファを固定サイズにしてしまって，プログラムがいつパンクするかが時間の問題

〔図1〕A君の解答

```
char b[100];
strrev(a)
char *a;
{
    int i, j ;
    j=strlen(a);
    for ( i=0 ; i<j; i++ ) {
        b[99-i] = a[i] ;
        b[100]='¥0';
        strcpy(a,&b[99-i]);
    }
}
```

〔図2〕B君の解答

```
strrev(a)
char *a;
{
    int i, j, len ;
    char temp ;
    len = strlen(a) ;
    i = len-1 ;
    for ( j=0; j<=len/2 ; j++, i-- ) {
        temp = a[j] ;
        a[j] = a[i] ;
        a[i] = temp ;
    }
}
```

〔図3〕D君の解答

```
char *strrev(a)
char *a;
{
    register char *up, *down, t ;
    for (   up=a, down=a+strlen(a) ;
            up < down ;
            up++, down-- ){
            t=*up;
            *up=*down;
            *down=t;
    }
    return a;
}
```

になっている．また，スタック上ではなくてプログラム外でバッファをアロケートしたので，A君のルーチンを使うプログラムのメモリがそのぶん食われる．

B君の解は，まだいいほうだが，Basicを連想させられる．B君はきっと，DIM文を書きたかったのだろう．コンパイラにもよるが，文字列の中の文字を添え字でアクセスするのは遅い場合があるし，プログラム全体がすっきりまとまらないことも多い．

C君，ご苦労さん．すごいこと知っているネ．しかし，この問題ではそんな技を披露する必要はない．十分なコメントをいれないかぎり，後からきた人はさっぱりわからない．レジスタの少ないマイクロプロセッサなら別かもしれないが．

D君は正解としよう．まず設計からみると，関数の戻り値が逆さまになった文字列へのポインタを返してくれるようにしている．これはありがたいだけでなく，多くのCライブラリの関数(strcpyなど)がこうなっているので，ポインタが返ってくるだろうと思い込むユーザがいるはずで，彼らは助かる．インデックス変数をregisterと宣言するのも常識である．また，文字のポインタ扱いは多くのコンパイラ，多くのアーキテクチャで最善の性能，コンパクトさにつながるだろう．細かい点だが，B君のプログラムだと，1文字だけの文字列でもその1文字を自分に上書きする．バグとまではいえないが，きれいでもない．

D君策は完全とはいえないが，ほかよりは間違いなくベター．さて，総評価に進む．

プログラマ	点数	時間	生産性
A君	20	1時間	20
B君	80	2時間	40
C君	75	2時間	37.5
D君	90	1時間	90

ここで生産性は，点数を消費時間で割って計算しているが，この基準でいくと，D君の生産性がA君のそれを4倍以上上回っていることになる．しかも不幸にもA君レベルのプログラマはそう少なくないと思う．本来A君のプログラムで将来でてくるバグをつぶす時間も計算に入れたら彼の生産性はさらに低い水準に落ちる．

生産性向上のアプローチと問題点

D君のようなプログラムを書けることイコール生産性が高くなるという方程式は必ずしも成り立たないとしても，D君的なプログラムを皆さんに組んでもらうためにはどうすればいいだろうか．

▶学校教育での対処と現実性

まず，大学での教育水準をあげることが頭に浮かんでくる．しかし社会人になってからプログラマに変身する人が多くて，大学では物理とか，工学とか，場合によっては文学専攻だったという人も多い．したがって，大学でこの問題に対応しようとすると，情報科学の部門だけでなく，全学生を対象とした大規模なソフト教育企画を展開しなければならないということになる．日本の大学はどうせ，そう急に真剣に勉強する場にはなってくれなかろう．

もし，学校にコンピュータ教育を導入するならば，小学校や中学校の時代からのほうがよろしいかと思う．で，ご存じの読者も多いかと思うが，こういった企画はいま文部省を中心に進行中である．しかし，市単位で教育政策を決めるアメリカに比べて，日本での教育は中央決定主義で有名で，何でも決めるのに何年もかかり，決めないうちに日本のソフトの遅れが悪化しかねない．この問題にかぎって，ある程度学校に権威を与えて，学校単位でコンピュータ教育を展開していってみて，その結果を見て全国のアプローチを決める，というのがよかろうかと思う．筆者も，高一のとき，独自にコンピュータ教育を手掛けてみようという恩師バンス先生のおかげで，コンピュータと出会うことが

できた．

▶**開発環境をよくするという考え方の危険性**

いや，教育の問題でなくて，開発環境だ，という人もいる．筆者の理解では，この発想がシグマ計画の出発点である．設計支援ツール，賢いコンパイラ，デバッガ，ドキュメンテーションのツール，さらにCASE(コンピュータ支援型ソフトウェア・エンジニアリング)システムというものさえあれば，という考え方である．

しかし，この方向は根本的に間違っていると思う．なぜならば，ツールは，必須の道具でありながら，それによる生産性の向上度はやはり利用者の腕前に左右される．まだ基本が備わっていないユーザだと，ツールはよくても邪魔，悪くて危ない場合さえある．悪い設計が早くできてしまう．悪いプログラムでもすぐデバグれてしまう*3．ひどいシステムのきれいなドキュメンテーションが簡単に作れてしまう．場合によっては，ツールの存在はソフト産業の諸問題を悪化する恐れさえある．

またツールは多くの場合，利用者が直接作るのが一番いい．ただし，もともと基本が抜けているプログラマたちだと，ツールを作る能力自体がない．また，高級なツールになると，その使い方も決して簡単ではなくて，かなりの勉強を要するので，体系的な考え方を身につけていないプログラマだと，ツールが良ければ良いほど使いこなせないだろう．結論をいうと，ツールは素人のレベルをあげるためのものではなくて，高いレベルに達した人々のためのものといえる．

現場教育の充実化

ソフト生産性の向上の秘密は，ソフト製造現場の教育性を高めることにある．ここで，どうすれば現場の教育性を高められるかの提案を一，二あげたいと思う．

▶**技術者間のコミュニケーションを高める**

日本的経営はなんでも，根回ししたり，ほかの人の意見を聞いたり，ちょっと関係人に説明するなど，コミュニケーションが大黒柱となっているといわれている．たとえば，日本企業の本社と海外支社との間のファックス量は，なんと外資系の企業のそれの十倍になっているようである．しかし，ソフト開発になると，一人一人端末の前に座り込んで，バグや問題がないかぎりコミュニケーションしないのではないかと思う．

一つのプログラムの骨組みがだいたい決まって，実際にコードる*4前に，簡単にプログラムの構造を2，3人で打ち合わせてほしい．これによって，お互いに教育しあえるのみならず，早い時期に致命的なロジックのバグが見出せよう．

また，プログラムが終わった時点ででも，また少人数でプログラム・レビューを行うメリットも大きいだろう．教育の意味もあるし，隣の人がプログラムの構造を知っておいたほうが，メンテナンスにも役に立つ．コーディングの前のレビューもその後のレビューも，いわゆるウォークスルーと呼ばれるものである．

▶**組織の問題**

つぎは組織である．きつい納期に追われながら開発しているとなると，チーフ・プログラマに当たる人は，チームの仕事をうまくこなすためにいろいろな問題を即決し，それを実施する権力がなければならない．日本の伝統的な企業組織構造ではこれが必ずしも可能というわけでもないので，ソフト開発用の新組織のアプローチを考える意味は十分にあろう．

また，日本にかぎった問題ではないが，技術的によくできる人が役職になってしまう傾向が強くて，せっかく一番貢献できるはずの技術者が現場から離されてしまうことがある．これに対応するために，偉くさせておいて現場に残すという考え方がある．いわゆる部下なし課長制である．アメリカでツートラック・システムと呼ぶこともある．すなわち，経営トラック，技術トラックのどちらかを本人が自由に選べる．この制度の導入によって，トップクラスのエンジニアが待遇面での犠牲なしに，効率よく現場に残り，後輩に教育を続けることができる．

日本の一部のメーカでも，こうしたアプローチを検討したり，導入に踏み切ったりしているようである．

＊

ソフト業界の将来のあり方の課題が山積している．そのうち，ソフト生産性の向上が急務である．皆さんが実際毎日仕事されているソフト製造現場にいながら，技術もどんどんレベルが上がっていくような，環境の教育性の高め方を考えて，生産性向上への第一歩を踏み出してほしい．

注：遅れている読者のため，新語4語を紹介する．

*1 **テスつ**：テストするとの意．例：テスった(テストした)，テスとう(テストしよう)，テスっていない(テストしていない)．

*2 **プログラむ**：プログラムを組む，プログラミングする．例：プログラもう(プログラミングしよう)，プログラんじゃった(プログラムをつくっちゃった)．

*3 **デバグる**：デバッグする，バグをとる．例：もっとデバグって，デバグっていない．

*4 **コードる**：コーディングする．例：もうコードろう，コードり終わった．

ロバート・T・マイヤーズ　㈱パシフィテック

ハード・ディスク・メンテナンス・ユーティリティ
SpinRite

秋元 潔

ハード・ディスクやフロッピ・ディスクは，入出力の速度をあげるためにセクタをとびとびに配置(セクタ・インタリーブ)しているが，自分が使用しているハード・ディスクのインタリーブ・ファクタが最適になっているかどうかを気にしたことがあるだろうか．16ビットのハード・ディスク・コントローラを使ったPC/ATでは2：1，8ビット系のコントローラを使ったPC/XTでは3：1のインタリーブ・ファクタになっていればOKといえる．IBM純正のマシンであれば，データ転送速度が最大になるようにインタリーブ・ファクタが設定されているので気にする必要はないが，部品を寄せ集めただけという感じのPC互換機を使用している場合は，一応チェックしたほうがよい．インタリーブ・ファクタを変更すれば，ディスク・アクセスがいままでより高速になることがあるからである．

ハード・ディスクのデータの読み書きは，目的のトラック上でセクタID(シリンダ番号，サイド番号，セクタ番号などの情報が格納されている)を読み出してIDフィールド内の情報が目的アドレスと一致したとき，その直後のセクタ・データ・フィールドに対して行われる．このため，ディスクにはあらかじめセクタID情報，その他の情報を記録しておく必要がある．この書込みを低レベル・フォーマットとか物理フォーマットと呼ぶ．これに対してDOSのFDISKコマンドで行う，ディスク・パーティションの確保，FAT(File Allocation Table)の作成，ルート・ディレクトリ・エントリの確保などは，高レベル・フォーマットとか論理フォーマットと呼んでいる．

ハード・ディスク装置は，ある程度のオフ・トラック(ヘッドとトラックの位置がずれること)が発生しても正常にリード/ライトできるように設計されているが，長期間使用しているとヘッド位置決め機構などの摩耗によりヘッドの中心とトラックの中心位置がずれてくる．その結果，図1に示したように，低レベル・フォーマットしたときに1回だけ書き込んだセクタIDと，必要のつど何回でも書込み(更新)されるセクタ・データのアライメントが不揃いになり，

“Sector Not Found”

を起こす確率が高くなる．

このエラーが発生しないようにするには，ヘッド位置決め機構が摩耗したあとの状態で，セクタIDとセクタ・データを書き直して両者のアライメントが揃うようにしてやればよい．つまり，摩耗したら摩耗したなりにヘッドの中心とトラックの中心位置を合わせてやろうというわけである．

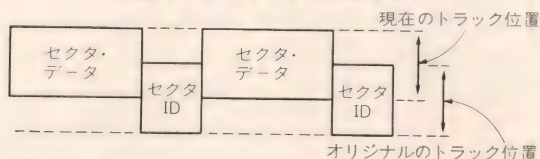
そこで今回は，貴重なユーザ・データを失わずにハード・ディスクのインタリーブ・ファクタを最適化したり，低レベル・フォーマットのやり直しができるGibson Research社のハード・ディスク・メンテナンス・ユーティリティSpinRiteを紹介しようと思う．

配布ディスクの内容

SpinRiteのソフトウェアは1枚のフロッピ・ディスクに収まっており，表1に示す内容である．

SPINRITE.COMはSpinRiteの本体プログラムで，つぎに示す機能をもっている．

(1) ハード・ディスクの表面検査
(2) 現在のインタリーブ・ファクタの表示，およびインタリーブ・ファクタが1：1，2：1，…，8：1のときのデータ転送速度の算出
(3) インタリーブ・ファクタの最適化(既存のデータを

〔図1〕トラックずれのようす

〔表1〕配布ディスクの内容

ファイル名	拡張子	サイズ	日付
SPINRITE	COM	43684	6-01-88
SPINTEST	COM	3002	6-01-88
PARK	COM	483	6-01-88
README	NOW	12733	6-01-88

破壊することなく任意のインタリーブ・ファクタに変更することができる．このとき，データ転送速度が最大のものを選べばインタリーブ・ファクタは最適化される）

(4) 既存のデータを破壊しないで，低レベル・フォーマットをやり直す

SPINTEST.COMは，ハード・ディスクが何回転すれば1トラックぶんのデータすべてをアクセスできるかという回転数と，そのときのデータ転送速度（バイト/秒）をつぎの形式で出力する簡易的なセクタ・インタリーブの診断プログラムである．

```
SpinTesting...
          x : revolutions to read a track,
    xxx, xxx : bytes transferred per second.
```

このプログラムはBasic言語で書かれており，そのソース・リストはパッケージといっしょに添付されてくる資料に掲載されている．SpinRite購入者でなくても，このプログラムを使用することは認められている．

PARK.COMは，ハード・ディスクのリード/ライト・ヘッドをシッピング・ゾーンに移動させる．電源をOFFするまえとか，電源ONのまま長時間ハード・ディスクのアクセスをしないときにこのプログラムを実行させてやれば，ディスクの表面とヘッドが接触してデータ記録領域を傷つけることがなくなる．

README.NOWには，SpinRiteを使用するときの注意や制限事項が載っている．T5100などの東芝ラップトップについては，SpinRiteがサポートできる低レベル・フォーマットと互換性がないので，SpinRiteは使用できないという注意書きがある．また，データ転送速度を速くすることのできる東芝用の特別なSpinRiteがあるらしいが，確認はしていない．

マニュアルはA5版で40頁ある．マニュアルを読まなくてもヘルプ・メッセージ，SpinRite Informationメニュー，README.NOWファイルを見れば，本誌の読者であれば十分使いこなせると思う．

必要最小構成

SpinRiteを実行させるには，IBM PCファミリ（XT，AT，PS/2，そのコンパチ機），バージョン2.0以降のDOS，1基のFDDとHDD，80文字/行のディスプレイ装置が最低限必要である．

SpinRiteは，MFM（Modified Frequency Modulation），RLL（Run Length Limited），ARLLなどのデータ記録方式をサポートしているので，ほとんどのハード・ディスクで使用できる．拡張ボード上にHDDを載せたPlus Development社のHARDCARDでは使用できないという制限が一部あるが，この制限は気にしなくてもよいと思う．詳しく調べたわけではないが，制限にひっかかるものは実行時に警告メッセージで知らせるようにしてある（らしい）からである．

ただし，SpinRiteで低レベル・フォーマットをやり直しても問題ないかどうかを一度確認する意味で，はじめてSpinRiteを実行するまえに，安全のためにハード・ディスクのバックアップをとっておいたほうがよいだろう．一度低レベル・フォーマットが正しく動作することを確認してしまえば，それ以降のバックアップは不要である．筆者は，ハード・ディスク1回転で1トラックぶんのデータをアクセスできる1：1のインタリーブ・ファクタをもつNorthgate Computer System社の386マシン，国産AT互換機（㈱マイクロリサーチのPC-ATLAS）で使用しているが，今のところ問題は発生していない．

コピー・プロテクトされたアプリケーションをハード・ディスクにインストールしている場合は，SpinRiteにかけるまえにそのアプリケーションをアン・インストール（uninstall：インストールの解除）しておく必要がある．そのままにしてSpinRiteすると，プロテクト用の特別な情報が書き込まれているセクタは“Bad Sector”として処理されてしまうので，そのアプリケーションはハード・ディスクから起動できなくなる．

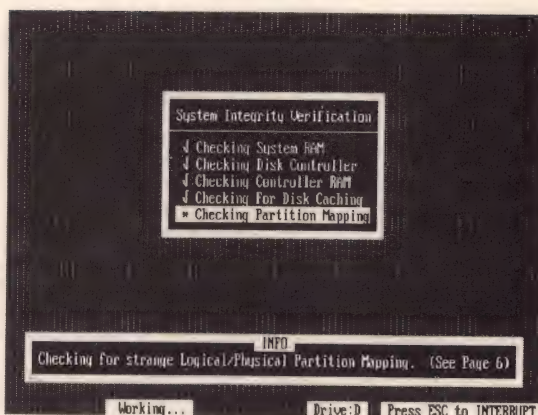
インストール

配布ディスクのプログラムはコピー・プロテクトされていないが，特別なブート・レコードが記録されているので，SPINRITE.COMをそのまま実行してもSpinRiteは起動できない．SpinRiteを起動するには，実行するマシンの環境をSPINRITE.COMに書き込む操作をする必要がある．つぎにこの手順を説明する．

① DOSのdiskcopyコマンドで配布ディスクのバックアップ・コピーをとり，これをインストール用のフロッピ・ディスクとする．

② インストール用フロッピ・ディスクをドライブAに挿入してCtrl＋Alt＋Delでシステムを立ち上げると，そのマシンの環境がSPINRITE.COMに自動的に書き込まれる．この書込みは，SpinRiteの特別なブート・レコードが行う．

③ format a: /sコマンドでDOSが起動できるフロッピ・ディスクを別に作り，このディスクに②で作成したSPINRITE.COMをコピーして，SpinRite起動用のシステム・フロッピ・ディスクとする．ディスク・アクセスなどのタイミングを正確にとる必要から，config.sys，autoexec.batにはディスク・キ

〔図 2〕メイン・メニュー

ャッシュ・プログラムやTSR(Terminate but Stay Resident：終了してもメモリに常駐しいつでも起動できる)プログラムの設定はしない．

SpinRiteを複数のマシンで使用する場合(個人が所有する複数のマシンでSpinRiteを使用することは許可されている)は，前述の手順でそのマシン専用のSPINRITE.COMを作成する．異なったマシンでインストールしたSPINRITE.COMを用いると，

"SpinRite Configuration Error"

と表示され，SpinRiteは実行できない．

ハード・ディスクの表面検査

ハード・ディスクの全トラックをスキャンして，その診断結果をトラック・マップ上に表示する．表面検査はリード・オンリで，ハード・ディスクへの書込みはしない．

SpinRiteを起動すると**図2**のメイン・メニューが現われるので，

1.Quick Surface Scan

を選択して検査したいディスク・ドライブを指定すると，RAM領域，ハード・ディスク・コントローラなどのシステム環境の状態をチェックする**図3**の画面が表示される．チェックは上から順に行い，チェックが終了した項目の左には√マークが付けられる．**図3**はディスク・パーティションのマッピング状況をチェック中であることを示している．

表面検査がはじまると，実行状況が**図4**のように示される．

Track Statusのc/Cは，SpinRiteで修復可能な未使用/使用中のトラック，u/Uは修復不可能な未使用/使用中のトラックを表すシンボルである．DOS経由では，リードしたデータがもともとエラーなしのデータ

〔図 3〕システム環境のチェック

〔図 4〕表面検査の実行状況表示

だったのか，ECC(Error Correction Code)によって復元されたデータなのかの区別はつかない．このため，SpinRiteはハード・ディスク・コントローラに直接アクセスして，修復できるデータかどうかの判断をしている．

TRACK MAPは，これまでにトラック番号0～717の718トラック分の表面検査をしたがすべてOKで，現在719トラック目を診断中であることを示している．そのほかについては図をみればわかると思うので説明は省略する．

参考までに，手持ちのNorthgate 386(クロック16 MHz，32 MバイトのHDD)と国産AT互換機(クロック8 MHz，20 MバイトのHDD)の所要時間を**表2**に示す．

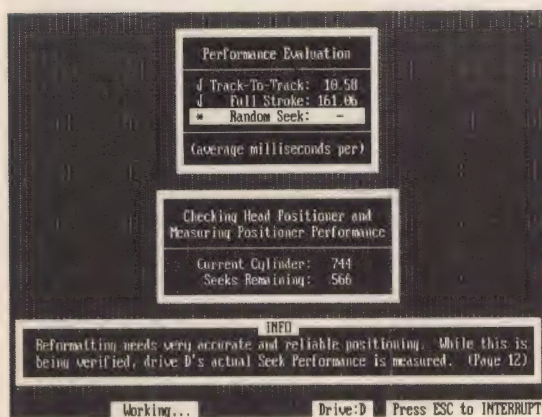
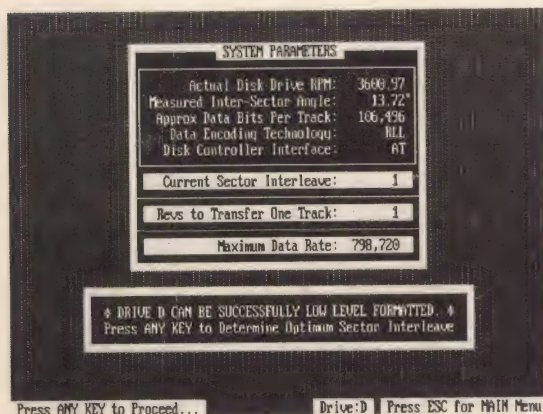
インタリーブ・ファクタの最適化

メイン・メニュー(**図2**)から，

2. Begin SpinRite Analysis
を選択すると，シーク時間の性能測定(**図5**)，セクタ転送タイミングの解析(**図6**)を行う．このあと，現在のインタリーブ・ファクタ(Current Interleave)，1トラックぶんのデータをアクセスするのに必要なディスクの回転数(Transfer Revs)，インタリーブ・ファクタが1：1〜8：1のときの平均データ転送速度(Avg. Data Throughput)，最適なインタリーブ・ファクタ(Optimum Interleave)が**図7**のように表示される．Enterキーを押してそのまま処理を進めれば，最適化されたインタリーブ・ファクタでセクタID，セクタ・データが書き直される．これ以降の動作は，後述する「低レベル・フォーマットをやり直す」の項と同じである．

〔表2〕表面検査の所要時間

	386マシン (32 MB)	AT互換 (20 MB)
表面検査	2分30秒	3分50秒

〔図5〕シーク時間の性能測定

〔図6〕セクタ転送タイミングの解析

なお，**図7**では現在設定されているインタリーブ・ファクタがすでに1：1の最適なインタリーブ・ファクタになっているので最適化する必要はない．ただし，矢印キーで別のインタリーブ・ファクタを選択してEnterキーを押せば，任意のインタリーブ・ファクタに変更することは可能である．この場合は，最適化されていないインタリーブ・ファクタで書き直されることになる．

シーク時間測定，セクタ転送タイミング解析の所要時間を**表3**に示しておく．

低レベル・フォーマットをやり直す

図7で目的のインタリーブ・ファクタを選択したあとEnterキーを押すと，低レベル・フォーマットの実行状況表示画面(**図8**)が現れてセクタIDとセクタ・データが書き直される．

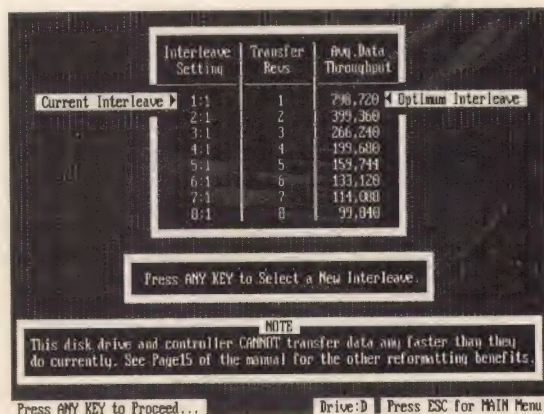
画面左上のD:はドライブ番号，1：1は**図7**で選択したインタリーブ・ファクタ，Depth:2は正常に書き込まれたかを書込み後に読み出してチェック(read after write)するときのパターン・テストの深さが2であることを示している．パターン・テストの深さは，低レベル・フォーマットを行うまえにつぎに示す4種類のなかから選ぶが，説明は省略する．

Depth1：書込み後の読出しチェックをしない

Depth2：1バイトごとに5種のパターンで読み出してチェックする

〔表3〕シーク時間測定/転送タイミング解析の所要時間

	386マシン (32 MB)	AT互換 (20 MB)
シーク時間測定	1分10秒	1分30秒
転送タイミング解析	30秒	30秒

〔図7〕インタリーブ・ファクタの選択画面

〔表 4〕低レベル・フォーマットの所要時間

	386マシン (32 MB)	AT 互換 (20 MB)
Depth1	5 分	9 分
Depth2	16 分	28 分
Depth3	110 分	105 分
Depth4	210 分	190 分

Depth3：1バイトごとに42種のパターンで読み出してチェックする

Depth4：1バイトごとに84種のパターンで読み出してチェックする

パターン・テストにひっかかった場合は，そのセクタにBADマークをつけてこれ以降使用できないようにしたあと，データはほかの使用可能なセクタに書き込まれる．

低レベル・フォーマットの所要時間を参考までに**表 4** に示す．信頼性の見地から通常はDepth3または4を指定するので，20 Mバイトでは約2～3時間かかる．

バッチ・モードでSpinRiteを動かす

SpinRiteは，コマンド・ラインからオプション・パラメータを指定してバッチ・モードで動かすこともできる．ここではコマンド・ライン・パラメータの説明は省略するが，たとえばカレントのハード・ディスク・ドライブの表面を検査したければ，

```
>A: spinrite autoexit batch quickscan
```

と入力する．

また，Depth2でインタリーブ・ファクタを最適化したければ，

```
>A: spinrite autoexit autoreport batch depth2
```

と入力すればよい．この場合，実行するまえにすでにインタリーブ・ファクタが最適化されていれば，インタリーブ・ファクタはそのままで低レベル・フォーマットをやり直したことになる．autoreportは，SpinRiteがどのような処理をしたのかのサマリ・レポートを自動的に生成する．サマリ・レポートの例を**図 9** に示しておく．

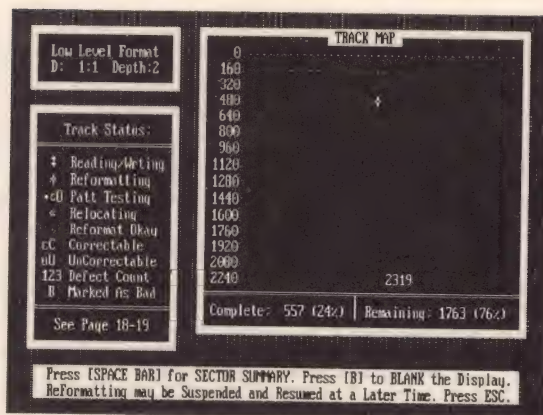
なお，SpinRiteを対話モードで動かした場合は，メイン・メニューから

4. Print Full Operation Summary

を選択すれば，**図 9** と同様なレポートが出力される．

*　　　　*

筆者が使用しているのはバージョン1.2，シリアル番号A0024793である．開発者(連絡先)・価格をつぎに

〔図 8〕低レベル・フォーマットの実行状況表示

示す．

開発：　Gibson Research Corporation
22991 La Cadena
Laguna Hills, CA 92653, USA
Tel ：(714)830-2200
Fax ：(714)830-0300

価格：　59ドル

配布ディスクにはSpinRite用の特別なブート・レコードが書き込まれているので，注文時には使用しているマシン名とフロッピ・ディスクのサイズ(5.25または3.5インチ)を指定したほうがよい．

おわりに

SpinRiteの開発者は，*InfoWorld* 誌のTech Talkのコラムを担当しているSteve Gibson氏である．このことと直接関係ないと思うが，SpinRiteの広告は筆者の知るかぎり，同誌に掲載されているだけである．

SpinRiteが最初に発売されたのは1988年2月である．筆者はそれからしばらくして購入したので，1年以上使用していることになる．低レベル・フォーマットは時間がかかるが，目安として3ヵ月ごとに行えばよいので使用するうえでの不便は感じていない．SpinRiteを使用した効果がどのくらいあるのかは確かめようがないが，低レベル・フォーマットをやり直すと精神的にすっきりした気分になれるから不思議である．

SpinRiteは，ハード・ディスクのアクセスに関する事故が起こるまえに事故を未然に防ぐPM(Preventive Maintenance：予防保守)用のソフトという性格が強いが，DOSで実際に読み書きできなくなったときのEM(Emergency Maintenance：故障保守)用にためしに使ってみる価値はあると思う．筆者は，残念と

〔図9〕サマリ・レポート例

```
******************************
*   Performance Evaluation   *
*----------------------------*
*    Track-To-Track:  10.58  *
*       Full Stroke: 161.06  *
*       Random Seek:  65.85  *
*----------------------------*
* (average milliseconds per) *
******************************


************* SYSTEM PARAMETERS ************
*+----------------------------------------+*
*|      Actual Disk Drive RPM:   3600.97 |*
*| Measured Inter-Sector Angle:    13.72'|*
*|  Approx Data Bits Per Track:  106,496 |*
*|    Data Encoding Technology:      RLL |*
*|   Disk Controller Interface:       AT |*
*+----------------------------------------+*
*   Current Sector Interleave:         1  *
*-----------------------------------------*
*  Revs to Transfer One Track:         1  *
*-----------------------------------------*
*            Maximum Data Rate:  798,720  *
*******************************************


**************************************
* Interleave | Transfer |  Avg.Data  *
*   Setting  |   Revs   | Throughput *
*------------|----------|------------*
*     1:1    |    1     |  798,720   *
*     2:1    |    2     |  399,360   *
*     3:1    |    3     |  266,240   *
*     4:1    |    4     |  199,680   *
*     5:1    |    5     |  159,744   *
*     6:1    |    6     |  133,120   *
*     7:1    |    7     |  114,088   *
*     8:1    |    8     |   99,840   *
**************************************


Page 1  - SpinRite(tm) Hard Disk Operation
          Summary -  Serial # A0024793
```

```
*********************  ****************** TRACK MAP ******************
*  Low Level Format  *  *    0 ........................................ *
*  D:  1:1  Depth:2  *  *  160 ........................................ *
*********************  *  320 ........................................ *
                       *  480 ........................................ *
*********************  *  640 ........................................ *
*    Track Status:   *  *  800 ........................................ *
*--------------------*  *  960 ........................................ *
*  |  Reading/Wrting *  * 1120 ........................................ *
*  *  Reformatting   *  * 1280 ........................................ *
* .oO Patt Testing   *  * 1440 ........................................ *
*  <  Relocating     *  * 1600 ........................................ *
*  .  Reformat Okay  *  * 1760 ........................................ *
* cC  Correctable    *  * 1920 ........................................ *
* uU  UnCorrectable  *  * 2080 ........................................ *
* 123 Defect Count   *  * 2240 ...................2319                   *
*  B  Marked As Bad  *  *                                               *
*--------------------*  *-----------------------+-----------------------*
*   See Page 18-19   *  * Complete: 2320(100%) | Remaining:    0 ( 0%)  *
*********************  *************************************************


******************* LOW LEVEL REFORMAT SUMMARY *******************
*        DOS Partition Status        | Clusters | Sectors |  Bytes    *
*------------------------------------+----------|---------|-----------*
* Marked Bad in the FAT Initially |   None   |  None   |   None    *
*.................................|..........|.........|..........*
*          Returned to Active Use |   None   |  None   |   None    *
*           Removed from Active Use |   None   |  None   |   None    *
*.................................|----------|---------|-----------*
* Marked Bad in the FAT Afterward |   None   |  None   |   None    *
*---------------------------------+----------+---------+-----------*
*  Partition's Total Defective Sector Count:         0  Sectors   *
*                   Data Relocated to Safety:         0  Clusters  *
*       Net Partition Storage Gain (or Loss):         0  Bytes     *
*******************************************************************


******** PRE-EXISTING DATA INTEGRITY SUMMARY *******
*  Error Implication and Nature: | Corrct | UnCorr *
*--------------------------------|--------|--------*
*  Sectors Containing Valid Data |  None  |  None  *
*   Sectors Not Currently In Use |  None  |  None  *
****************************************************


Page 2  - SpinRite(tm) Hard Disk Operation Summary -  Serial # A0024793
```

いおうか，幸運にもハード・ディスクの事故に遭遇していないので，これ以上のコメントはできない．

SpinRiteの実行中は，いつでも実行の中断ができるようになっている．中断して終了した場合は，次回の起動時に中断したところから再開する(Resume機能)か，前回の作業を白紙にして新しくやり直すかを選ぶことができる．つまり，ムダな作業を2回やらせないような配慮がされている．詳しい説明は省略するが，SpinRiteを使用して感心したことはこの気配りであった．もし，筆者が設計していたとしたらResume機能は考えつかなかっただろう．技術者は，ユーザの立場にたった使いやすく気配り十分な仕様を考えつくのは苦手である．筆者が米国PCソフトに興味をもっているのは，本業であるソフトウェアの検査・評価の参考にするためであるが，また一つ勉強になった．

同じジャンルのソフトとして，Kolod Research社が開発し，Paul Mace Software社から発売されているhTEST/hFORMAT ver 2.0も購入したが，パラメータの数が多いのと使い心地がよくないので一度使ったきりでサヨナラした．

あきもと・きよし　(有)プロソフト アソシエイツ

RISC CPU メーカ 4 強の主流争奪戦

1990 年代を目前とした現在，OA 市場の主役変更が胎動している．PC は WS へ，OS は Unix へ，そして CPU は RISC へ，と．そして，CPU メーカには，今，千載一遇といわないまでも，10 年に一度の好機が訪れようとしている．急成長が約束されている 32 ビット Unix ワークステーション(以下 WS)向けの RISC CPU 市場が，大きく開かれようとしているからである．しかも，その RISC CPU 市場参入メーカ系列は，すでに 4 社 —— Sun Microsystems，Mips，モトローラ，インテル —— に絞られている(表参照)．

1980 年代のコンピュータ(OA)市場は，IBM PC(とその互換機)が主役の時代であった．それは，また，マイクロソフト社製の MS-DOS を OS とし，インテル社製の 8086，80286，80386 を CPU (汎用 CISC) に採用した PC の時代であった．この市場は，今や，累積設置台数 3,600 万台の市場と成熟し，その CPU 供給元となったインテル社の業績(年商)も，7 億 8,900 万ドル(1980 年)から 29 億ドル(1988 年)へと 3.7 倍増の飛躍ぶりである．

1990 年代は，PC が WS へ，そして，CPU は，CISC から RISC へと主役が移行する．RISC WS の市場規模は，今後 5 年間で，10 億ドル(1988 年)から 100 億ドル(1993 年)へと 10 倍増する一方(図参照)，RISC CPU 市場は，1,500 万ドル(1988 年)から，2 億 5,000～13 億ドル(1992 年)へと，さらに大幅な急成長をする勢いをみせている．

RISC CPU 優勢の風は，3 方向から吹き始めている —— (1) 主力コンピュータ・メーカが，RISC CPU 採用を決定していること，(2) ソフトウェア開発会社が，こぞって RISC アーキテクチャ・ベースのアプリケーション・ソフト開発に着手していること，(3) VAR (OA 流通) が，RISC・WS システムを要望し始めていること．

この風の流れの下，1980 年代(CISC 時代)の 2 巨頭 —— インテルとモトローラ —— にかわって，1990 年代の新主役 2 社 —— Sun Microsystems と Mips Computer Systems —— が登場した．新旧の際立った違いは，前者の場合，汎用の CISC CPU 2 大メーカができるだけ独占 (セカンド・ソースを形成しない)供給体制をとったのに対し，後者の場合 (Sun，Mips が，自ら半導体メーカではないことも事実であるが)，オープン・アーキテクチャ (非独占/セカンド・ソース) 化を拡大実践していることである．

さらに，WS 専用 RISC CPU 市場では，Sun と Mips が先発であり，インテルとモトローラは後発メーカである．しかも，後者 2 社は，今なお，「本業」が汎用の CISC CPU メーカであることから，RISC (まさにリスク) に賭ける執念に，今ひとつ不透明な部分が残っている．そこで，先発・後発両陣営の RISC CPU 市場戦略には，興味深い差異が出てくる．Sun と Mips 側が，ユーザのエンジニアを対象として，自らがもつソフトウェア資産やアーキテクチャの優位性を説く，いわば，「ボトムアップ作戦」に対して，インテルとモトローラは，1980 年代主流の CISC CPU 資産の継承力を，ユーザの経営陣に対して訴えかける，いわば「トップダウン作戦」が展開されている．いずれも，「RISC に賭けている CPU メーカ」を売り込んでいるには違いないが，実績で先行する新興 2 社が，現状，有利な情勢といえよう．

いずれにせよ，RISC CPU メーカの対ユーザ売込み合戦は，即，「ウチが主役(主流)」の認知促進が重要なカギとなる．そこで展開される戦法は，「主力ユーザを何社味方につけたか」という，カスタマ・アライアンス(顧客獲得戦略同盟) と化す．すでに，Sun は，AT&T，Xerox，Unisys，ICL，Solbourne (松下)，東芝，セイコー電子など，Mips は，DEC，ソニー，日電，Silicon Graphics，Ardent など，モトローラは，Data General，NCR，Tektronix，三洋電機，立石など．そして，インテルは，Olivetti，Stratus，(IBM ?) などのユーザを，それぞれ獲得している．

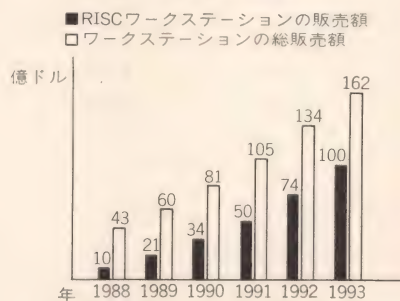
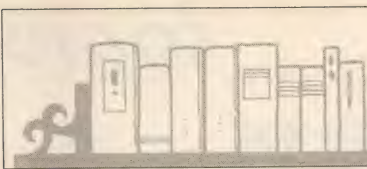
*　　　*

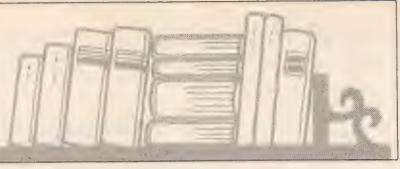
RISC CPU 市場の大型形成が注目されるのは，WS 市場の主力 CPU としての意義は，もちろんある．1988年 32 ビット RISC CPU ベースの WS 販売台数は，59,000 台であったのが，5 年後の 1993 年には，650 万台と 100 倍増以上の規模に達し，32 ビット OA コンピュータ市場の 41% を占めるに至る，といわれている (Stanford 大学調査)．さらに，RISC CPU が，戦略的 (そして，日本に重要) な意義をもつのは，周辺の ASIC やメモリ・チップ産業が，CPU 市場の 5 倍規模で形成されていくことである．RISC CPU の波及効果は，じつに，広大で深遠に展開されていく (RISC CPU 関連記事，1988 年 3 月，4 月，7 月，10 月，11 月各号の拙稿を参考にされたい)．

内田登美雄　ジャーナリスト

〔表〕 RISC CPU メーカ 4 社の営業戦略

メ ー カ	RISC CPU	主戦略とセールス・ポイント
Sun Microsystems	SPARC	● SPARC の大規模ソフトウェア資産 ●「PC のつぎの OA コンピュータ市場標準は，SPARC WS 以外ない」
Mips Computer Systems	R2000 R3000	● Mips 独自のアーキテクチャと開発ソフトウェア・ツールの優秀性
Motorola	88000	● WS メーカへの安定した CPU 供給力 ●「68000 のつぎの標準は，Motorola 製 RISC CPU しかない」 ● 88 Open(ソフトウェア/システム開発会社のコンソーシアム) の展開
Intel	860	● グラフィック・アプリケーションにおける差別化 ●「IBM コネクション(386，486 のつぎは 860)」の強調展開

〔図〕 RISC ワークステーション世界市場規模の推移

(注) 1988年は実績，1989年～1993年は推定．
資料:Dataquest社

いきあたりばったり

読書日記

×月×日（土）

書泉ブックマートに，角川文庫のリバイバルコレクションが，すこしのこっているという情報があったので，昼前から神保町までかけつける．『キリシタン版エソポ物語』や『江戸名所図会』の後半3冊などを，やっと手にいれることができた．最初にリバイバルコレクションが書店にでたとき，それをおしえてくれたNさんにも，電話でつたえておいたら，書泉ブックマートにあらわれたので，ビアホールのLで一緒に昼食．生ハムや自家製のソーセージなどで，生ビールをのむ．

×月×日（月）

夕方からS社のN社長と，大塚のEで一献．先日S社からでた本にエッセイをかいたのだが，その本を何冊かおくってもらおうと電話したところ，直接わたすからあいましょうということになってしまった．要はせっかくだから，一緒に酒をのもうということなのだが．相鴨のつくねが，なかなかうまい．そのあと早稲田のKにでかけ，ひさびさに仔牛の脳味噌の唐揚で，葡萄割をのむ．

×月×日（水）

今日で会社が創立10周年となる．1979年の今日，正式に会社として登記したのだった．よく10年もつづいたものである．

午後から用事があって外出．ついでに書店により，

三矢直城，田中一男『C言語による実用ファジィブック』ラッセル社　2,718円

など7冊．ラッセル社は機械翻訳やエキスパート・システムといった分野で，ソース・プログラムつきの興味ぶかい本をだしているが，これもその一冊．前半ではファジィ集合論やファジィ推論などについて，理論的な説明をおこない，後半でコマンド・インタプリタ形式で記述された，ファジィ理論処理システムが，しめされる．処理システムはMS-DOS上でうごくようにMS-Cでつくられ，ディスク・サービスもおこなわれているが，20,000円とかなりたかい値段がつけられている．

×月×日（金）

早稲田のN社から，一太郎のバグフィクス版がとどいたから，インストールしてくれといってくる．昼食を御馳走してくれるというので，昼前にでかけて実行用のディスクを作成．昼食はいくらいりのわっぱ飯にビール．

×月×日（土）

でかけようとしたら，荷物がとどく．創立10周年の記念品としてあつらえた，“10 Years from Thursday”というオリジナル・ブランドのハウス・ワインである．ラベルにはタミール文字の「10」をあしらってある．ちょうど会社を登記したのが木曜日だったので，それにちなんで名前をつけた．何本かは製造元に名簿をわたして直送してもらっており，とどいたのは直接手わたししたりするための分だが，それでもかなり場所をとる．

×月×日（月）

Computer Music Journal のVol. 13，No.2がとどく．オブジェクト指向にもとづく音楽システムの論文があつめられている．Smalltalk-80によるものの他，NeXT上のシステムについての論文も2篇ある．

×月×日（火）

「UNIXシンポジウム」の初日．本当は9時30分からなのだが，ついたときは10時30分をすぎており，OpenWindowsについての発表がはじまっている．そのあとOSF関連の発表が二つあって，午前中のセッションは終了．展示会場の方へいったところ，NeXTのブースに人が殺到している．丸善もUnix関係の洋書の展示をしていて，Hさんに声をかけられる．

午後の最初のセッションはAT&Tからの報告なので，ロビーで雑談をしてすごす．今回は終日，コーヒーとつめたい水のサービスがあるので，ロビーのいごこちがいい．最後のセッションは日本語入力関係で，これは真面目にきく．

Proceedings をひらいたところ，中に『GNUダイジェスト』という小冊子がはさまっていた．*GNU's Bulletin* の翻訳だそうで，GNUプロジェクトの最新の情報がつまっている．それによると，なんと匿名希望のイギリス人が，Free Software Foundationに10万ドルの寄付をしたとのことで，かなりおどろかされた．

夜は旧友のTとひさびさにあって，日本橋の洋食屋Tで一献．ハムサラダや牛舌芥子漬で生ビールをのむ．

×月×日（水）

「UNIXシンポジウム」2日目．今日もすこし遅刻してしまう．午前中はウィンドウ関係のセッション．午後の最初のセッションではCASEや4GLといったあまり興味のない話があったので，その間はロビーで昼寝．最後の通信関係のセッションで，シンポジウムはおひらきとなる．

×月×日（木）

月もなかばにさしかかったので，そろそろ決算の準備にかからなければならない．午前中は銀行を4軒まわって，残高証明などを手配する．

×月×日（金）

夕方から銀座の洋食屋Rの座敷で，会社の創立10周年記念の宴会．Rはポークカツレツや冬場のカキフライが絶品の，昔風の洋食屋なのだが，コースをたのんでおいたところ，かなり本格的なフランス風料理がでてきて，堪能する．

×月×日（土）

昨夜，宴会のために京都からでてきたN君につきあって，昼前から秋葉原にでかける．とおりがかりの店で，カシオのDigital Hornを衝動買．N君はMacintoshのソフトウェアが目的で，いろいろとかいこんでいく．昼食は室町の蕎麦屋Sで天もり．N君とわかれたあと一度帰宅し，あらためて神保町にでかける．

×月×日（火）

銀行へでかけたついでに書店によ

祐安重夫

り，７冊．

J.E. レイピン（矢吹道郎他訳）『ポータブル UNIX プログラミング――システム間の互換性』啓学出版　2,913 円

は，ちょうど２年前の「読書日記」で紹介した，

J.E.Lapin, *Portable C and UNIX System Programming.* Prentice-Hall.

の翻訳である．Unix の各バージョンの間の相違点を網羅し，移植性のたかいプログラミングをおこなうためのガイドブック．

×月×日（水）

昼からうちあわせのため外出．電車のなかで，

春木良且『オブジェクト指向への招待――思考表現のための新しい技法』啓学出版　1,942 円

をよみはじめる．著者は富士ゼロックスに所属しており，Smalltalk-80 を中心に，オブジェクト指向のかんがえかたが，まとめられている．

×月×日（木）

S 社から電話がかかってきて，NeXT をかりないかとのこと．こういう話は，即座に承知する．もちろん原稿をかくという条件つきである．

夕方から N 社の N，Y 両氏と I 嬢が来訪．N 社で Macintosh の雑誌の編集をひきうけるという話があり，その相談である．話の後は到来物の生酒で一献．茄子の塩もみにこんにゃくの煮付，焼茄子に山女の塩焼，そうめんなどをあつらえたところ，I 嬢に「小料理屋みたい」といわれてしまった．

×月×日（日）

参議院議員選挙．かんがえたすえ，比例代表区は「みどりといのちのネットワーク」に投票する．このネットワークの事務所は，神保町のすずらん通りにあり，無農薬野菜などの販売をやっていた．これとまぎらわしい名前の，西ドイツの DIE GRÜNEN の日本版と誤解されそうなミニ政党が別にあるのだが，そちらの方の名前はポルポト派の「緑の革命」を背景としているらしく，エコロジー運動とはまったく異質のもののようだ．言葉のつかいかたを非難するつもりはないのだが，わざと誤解をまねくような，物ほしげな名前をつけているという感じもしないではない．

×月×日（月）

昨夜から徹夜で，昼前までかかって１年分の帳簿をまとめる．午後から N 会計事務所の K さんがやってきて，夕方までにどうにか決算のめどがつく．

×月×日（火）

今日から毎年恒例の，銀座のイエナ洋書店のバーゲンがはじまるので，昼前からでかける．今回は英語版の暗黒舞踏の本や，映画・演劇関係など６冊で 9,300 円．

×月×日（水）

朝９時前に水道橋まででかける用事があったので，それがかたづいたあと，10 時まで喫茶店で時間をつぶす．本屋が開店するのをまって，11 冊．

大野義夫編『TEX 入門』共立出版　2,670 円

がでている．1987 年から 1988 年にかけて，『bit』に連載されたものに，大幅に加筆がおこなわれている．TEX の全容に関する日本語の本は，これが最初ではないだろうか．難点があるとすれば，実際に TEX を文書作成に利用することよりも，TEX の内部構造や概念といったことに，説明の重点がおかれているきらいがあり，かならずしも利用者むけの本とはいえなくなっているところである．また最近では日本語対応の TEX がそれなりに普及しつつあるが，そのあたりの説明がもうすこしほしいところである．次の段階として，LATEX を中心とした，実際の文書作成の参考になる本が，のぞまれる．

興味ぶかいのは，この本自体が日本語化された LATEX でつくられていることである．版下の出力は大日本印刷でおこなわれているので，本としての体裁とできばえは，普通の本と同等かそれ以上である．ただ，著者たちが入力した原稿を，そのまま利用しているので，キーボード入力時のあやまりを，直接反映したとおもわれる誤植が，非常におおい．このような形式で本がつくられるようになった場合，編集者がどのようにそこに介入していくべきかが，大変おおきな問題である．編集者にも，TEX を利用できるだけの能力が要求され，出版社もそれに対応する設備をもつ必要がでてくるのではないだろうか．

×月×日（金）

夜の９時 30 分頃，NeXT がとどく．８月末までかしてくれるとのこと．さっそくセットアップしてうごかしはじめて，ふと気がついたら深夜になっている．

×月×日（土）

例によって昼から神保町にでかけ，19 冊．

岩谷宏という名前は，最近コンピュータ雑誌でよくみかけるが，コンピュータに興味をもった音楽業界の人間だとおもっていたら，

岩谷宏『ラジカルなパソコン入門』筑摩書房　1,000 円

の著者略歴に，「編集者・音楽プロデューサーを経て，現在，パソコン関係の評論家・翻訳家・プログラマとして活躍中」とかかれており，いささかおどろいた．この本はそのような著者によるパーソナル・コンピュータの入門書で，技術的にはこまかいあやまりをふくんでいるが，それをおぎなってあまりあるだけの，過激な思想性がこめられた，おもしろい本である．ある意味では独断と偏見にみちた著者のかんがえかたに，全面的に賛同しようとはおもわないが，毒にも薬にもならない凡百の「パソコン入門書」と称するものにくらべれば，はるかにまっとうな本といえる．

（消費税反対の立場から，和書の価格は本体価格を明示しています）

CD-ROM 標準化へ

日本電子出版協会(JEPA)は7月18日，CD-ROMの標準化テスト・ディスク「和同開珎」開発に成功．CD-ROMの標準化に関しては昨年，論理フォーマットの国際規格ISO 9660が制定されたが，そのなかでは，日本語利用の具体的な取り決めまでは盛り込まれていない．そのままでは混乱を引き起こす懸念もあるとして，JEPAが日本語の利用法についての推奨案を作成・提案し，あわせてPRディスクとなる「和同開珎」を開発した．　(各紙7・19)

◈

CD録音1回だけOK…著作権保護の問題を背景に世界的に調整が難航していたDAT(ディジタル・オーディオ・テープレコーダ)について，日米欧のハード，ソフト業界は，ディジタル複製を1回のみとする方式を採用することで合意した．新規格は「シリアルコピー・マネージメント・システム」と呼ばれる．今後，ソフト業界および日欧のDATメーカ15社は，新方式を各国の政府に提案，技術基準を統一したうえでソフト，ハードの開発，販売を行っていく．　(各紙7・29)

◈

ネットワークに資格制度…郵政省は情報通信ネットワークのエンジニア人材を育成するため，今秋早々にも関係者による推進協議会で具体的検討を始め，できれば来年中にも「ネットワーク・アーキテクト」の資格制度を発足させたい考えである．資格制度はネットワーク構築・運用などの実務を担当するスタッフ・ネットワーク・アーキテクトと，総合的な管理を行うチーフ・ネットワーク・アーキテクトの2段階の資格とし，各種の企業ニーズに応じた人材確保を支援する制度．

(電波，日刊工業7・25)

◈

VCCIへの届出件数…VCCI(情報処理装置等電波障害自主規制協議会)への機器の届出件数は1988年度に3822件．ノイズ規制は1986年6月から実施されたが，適合確認届出件数は年々増加，1988年度は前年度比81％増となっている．昨年末から従来からの規制緩和措置が解除され，マージン・ゼロの最終段階に入っており，適合確認届出が新局面を迎えた．　(電波7・26)

参院選でも活躍したパソコン通信

政治にかかわる意見の表明手段としてパソコン通信が注目されている．時間や距離の制約がなく，同時に多人数にメッセージを送れる機能は選挙運動にもうってつけ．ある大手のネットでは政党本部の事務局員が消費税やリクルート事件の意見を求めたり，党の見解を伝える手段に利用している．別のネットでは複数の政党のスタッフが司会役を引き受け，「オンライン・デモクラシー」と題した電子会議も開かれている．

(読売夕刊7・10)

◈

ハイテク戦法に困惑…ニューメディアに関しては公選法にも明確な規定がなく，自治省は新手の出現に頭を悩ましている．自治省によると，現行の公選法でも不特定多数に向けての，ファクシミリによる文書の送信やCATVでの投票依頼は明かな選挙違反．しかしパソコン通信などについては「有権者全体に対する浸透度合がまだ低い」として当面は静観のかまえだ．(日本経済夕刊7・18)

◈

マーケッティングに活用…放送や雑誌のメディアがパソコン通信を使って読者，視聴者のニーズを探るケースが増えている．「ニフティ・サーブ」は会員5万2,000人で140のフォーラムがあるが，最近はこのフォーラム全体のアクセス量の過半数が，メディアミックスと呼ぶ放送や雑誌関係に集中してきている．こうした傾向について中村明NIF社常務は「市民の要求しているのは事実の情報ではなく経験の情報．パソコン通信は経験情報が飛び交う世界で，それを人が求めているとすればそれを取り入れられるかどうかが視聴率や本の売行きを左右することになる」と解説する．　(日刊工業7・12)

◈

NTTのBBS，86局移管し，15局廃止…NTTは101の直営パソコン通信局(BBS)のうち，86局を企業・団体に運営を移管し，15局を廃止した．NTTはパソコン通信の普及を目指して無料サービスをしていた．郵政省は今年3月，「無料のBBSは有料サービスをしている民間事業者との公正競争に反する」「電気通信事業法によりパソコン通信サービスをするには郵政省の許可が必要」などと指摘．この結果，NTTは6月末ですべてのBBSを閉鎖した．

(日経産業7・15)

IBMをめぐるパソコン戦略

OS/2にイーサネット追加…IBMは，ネットワーク・ベンダ3社(3コムコープ社とアンガーマンバス社，ウェスタン・デジタル社)とライセンス契約を結び，OS/2拡張版にイーサネットLANプロトコルを加えた．契約の結果，ネットワーク・ドライバ・インターフェース仕様(NDIS)にしたがうことになるため，OS/2拡張版1および2は，3社のイーサネット・アダプタをサポートできるようになる．NDISは，3コム社とマイクロソフト社が共同で開発したLAN用デバイス・ドライバ仕様である．　(電波7・10)

◈

4M DRAMチップを採用…IBM社は，同社が開発した4MビットDRAMを使ったパソコン用メモリを開発，PS/2パソコンの拡張メモリ・ボードとして売りだした．今年いっぱいは限定販売とするものの，年明けからは同社の全商品に4MビットDRAMの使用を拡大，本格販売することにしている．

(日本経済夕刊7・26，電波7・27，日刊工業7・28)

◈

ハード・ディスクを実質値下げ…日本IBMは，PS/55で採用したOS/2の普及を促進するため，ハード・ディスクの事実上の値下げにふみきった．同社はすでに増設メモリの値下げをしており，ユーザがOS/2への移行をしやすい環境を整えることで，OS/2とPS/55の拡販を目指す．OS/2を利用するには増設メモリのほか，60Mバイト程度のハード・ディスクが必要．ユーザが現在所有しているハード・ディスクとの差額を払うだけで，60Mバイトのハード・ディスクを供給するようにした．

(日経産業7・25)

◈

情報ファイル　ニュース・ダイジェスト

IBM 互換機高速パソコン…米のIBM 互換機メーカ2社が高速パソコンを相次いで売り出す．コンパック・コンピュータ社の「デスクプロ386/33」と AST リサーチ社の「プレミアム 386/33」．いずれもインテル社の 80386(33 MHz)の CPU を使った．IBM は高速パソコンでは互換機メーカの食い込みを完全に打ち切るため，AT 互換でない新パソコン PS/2 を作った．高速バスはマイクロチャネルで PC/AT の周辺機器は使えない．互換機メーカは PC/AT ソフトを使える高速パソコン用バスを作ることにして，EISA という組織を作った．　(日経産業 7・10)

ラップトップ型が鈍化

ラップトップ型パソコンの売行きが急速に鈍化してきている．7月に入り東芝などが A4 判サイズの超小型「ブック型パソコン」を発表したことで，新製品の発売待ちによる一般消費者の買い控え傾向が目立っている．実売価格も荷余り感の強い日本電気製品を中心に下がっている．日本電気の「PC-9801LV22」(標準小売価格 37 万 8,000 円)の実売価格は6月始めに比べ2万円前後下がり，セイコーエプソンの「PC-286LE-STD」(同 36 万 8,000 円)も 25 万円代を割るケースが目立っている．

(日本経済 7・27)

◈

パソコン輸出規制緩和…モスバッカー米商務長官は，パソコンの輸出規制を緩和すると発表した．規制緩和によると，商務省の輸出承認基準対象をこれまでの演算処理速度・毎秒 650 万ビット以上から同 6800 万ビット以上に引き上げる．具体的には，IBM 社の AT モデルと AT 互換機について，輸出認可を廃止する．米政府は，この規制緩和をソ連圏，中国向けにも適用する．

(日本経済夕刊 7・19，電波 7・21)

◈

光電子 IC の共同開発…モトローラ，ヒューズ，ハリスなど米電機・半導体の大手7社が光電子集積回路(OEIC)を研究開発する企業連合を組む．光電子 IC は一つの基板上に半導体レーザ，受光センサと電子回路などを組み込んだ新型素子．現在の光通信や光情報処理システムでは，光関係の部品と電子回路を別に作って組み合わせる．これを一体化すれば，通信演算速度が上がり小型で信頼性の高い部品ができる．

(日本経済 7・15，電波 7・17)

企業内教育へ CAI 導入

企業内教育へ CAI の導入が急進展している．日本能率協会が今年4月，530 の企業の教育関連部門を対象に「企業内教育における CAI の現状と将来」調査を実施した結果をまとめ発表したが，これによると 1986 年 11 月に同じテーマで実態調査したものと比較し，CAI 導入企業は 6.1％(37 社)だったのに対し，今回は 22.5％(119 社)へ，3.5 倍と飛躍的に増えており，今後ますます CAI を使用して企業内教育をする企業が増えていくと，同調査は指摘している．

(電波 7・29)

◈

ソフト開発の機械化…通産省は，将来のソフトウェア人材不足を抜本的に解決するため，平成2年度から8ヵ年計画で総額 50 億円を投じ「次世代協調型アーキテクチャー(基本設計思想)」の研究・開発に着手する．ソフトのメンテナンス，要求分析，論理情報の入力など，これまで人手に頼っていた知的作業を機械化するのが開発のねらい．具体的には IPA (情報処理振興事業協会)内に「国際ソフトウェア基礎技術研究センター」(仮称)を設置し，産官学共同で研究・開発に取り組む．

(日刊工業 7・26)

スーパコンピュータ業界に新顔

新旧勢力が入り乱れて開発競争が続くスーパコンピュータ業界に，また一つ新顔が登場した．米国のコンピュータ・グラフィックス専用装置メーカのエバンズ&サザランド・コンピュータ社(ユタ州)．同社のスパコンの名称は ES-1．競合各社が注目しているのは処理速度(1600 MIPS)と採用した OS “Mach” の使い勝手だ．Mach はカーネギーメロン大学が国防予算を得て開発した OS で，複数 CPU をコントロールでき，しかもバークレー版 Unix の 4.3 BSD と互換性がある．S. ジョブス氏の NeXT 社も新 WS に採用しており前評判は高い．　(日経産業 7・14)

◈

中古スパコンが東北大へ…米クレイ・リサーチ社は日本メーカの大学向け大幅値引きに対抗するため，値段の安い中古スパコンを使って商談を進める．第1弾として東北大学向け商談を検討している．米通商代表部は外国製スパコンを事実上，閉め出す不公正な慣行だとして，日本の政府調達を包括通商法スーパー 301 条の適応対象とした．しかし，政府間協議の決着までには時間がかかるため，中古機を使うことで価格競争にのぞむことにした．

(日本経済 7・31)

◈

韓国がスパコンを開発？…韓国の政府系研究機関が「米国，日本に続いてスーパコンピュータの開発に成功」と発表，その中身が物議をかもしている．開発したのは基礎技術の一つにすぎず，スパコンの開発成功というには早すぎるためだ．

(日経産業 7・28)

◈

BiiN 日本へ進出…シーメンスとインテルの折半出資で 1988 年に発足した合弁会社 BiiN が今秋にも日本市場に参入する．同社の並列コンピュータは，CPU にインテルが BiiN 専用に設計した 80960 を複数組み合わせている．OS には Unix システム V．もともと CIM 用など FA コンピュータとして開発したが，フォールトトレランス機能，リアルタイム処理機能，セキュリティ機能などを生かして金融などのオンライン処理分野でも大きな需要を見込んでいる．

(日経産業 7・20)

◈

光ニューロコン実証に成功…三菱電機は，中央研究所において世界で初めて「アルファベット 26 文字を認識する光ニューロ・コンピュータ」の実証に成功．同社では，主要部分を光素子で構成し，光による情報処理に適した「ディジタル学習アルゴリズム」を考案して採用したことが成功したキーポイントであるとしている．この光ニューロ・コンピュータは，発光素子アレイ，ディジタル的に動作する光配線・結合素子(空間光変調器)および受光素子アレイで構成されている．　(各紙 7・28)

カラー液晶ラップトップ・パソコン

日本電気および日本電気ホームエレクトロニクスは，PC-9801シリーズの新製品としてカラー8色の表示が可能なラップトップ・パソコンPC-9801LX5Cを発売した．同製品はPC-9801LX2/4/5(白黒液晶ディスプレイ)の上位機種であり，以下のような特徴をもつ．

(1) バックライト付き640×400ドットSTN(Super Twisted Nematic)カラー液晶ディスプレイを採用したことにより，ドット単位で白，緑，赤，青，水色，紫，黒の8色カラー表示が可能(アナログRGBディスプレイ接続時は4096色中16色の表示が可能)．

(2) CPUに80286(クロック12/10MHz切替え)を採用し，メモリ・アクセスはノー・ウェイト．

(3) CPUにはV30(8MHz)も搭載しており，従来のPC-9801シリーズとの互換性を保持している．

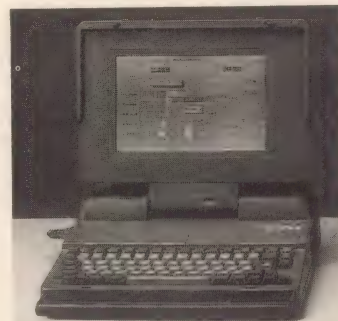
(4) 1Mバイト・タイプの3.5インチ・フロッピ・ディスクを2台内蔵．

(5) 40Mバイトの3.5インチ・ハード・ディスク(平均シーク時間28ms)を内蔵．

(6) ユーザ・メモリとして640Kバイトを標準実装し，最大3.6Mバイトまでのメモリを本体内に拡張可能．

(7) 3種のカスタムLSIを採用し，本体サイズは339(W)×380(D)×115(H)mm．

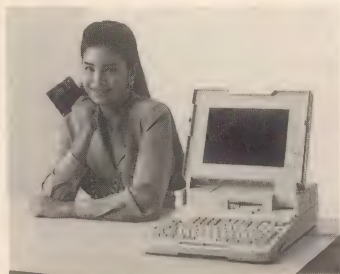
(8) 日本語OS/2，PC-UX/V(2.0)に対応し，日本語MS-DOS ver 3.3 A/3.3においてEMSの利用が可能．

(9) 本体と一体化して利用する専用拡張アダプタ，あるいはPC-9801シリーズ用拡張ボードを3枚内蔵できるI/O拡張ユニットが使用可能．

電源はAC 100 V±10％，消費電力は90 W(最大100 W)，本体重量は8.7 kg．

【価格】PC-9801LX5C(本体)：74万8,000円

日本電気㈱
〒108 東京都港区芝5-33-1
NECパソコン・インフォメーション・センタ　東京：03(452)8000
大阪：06(943)9800

ラップトップ・ワークステーション

日立製作所は，パーソナル・ワークステーション2020シリーズのラップトップ・モデルとして，高精細液晶ディスプレイを採用した2020モデルLを発売した．おもな特徴はつぎのとおり．

(1) バックライト付き1120×780ドット高精細白色液晶ディスプレイを採用し，24×24ドットの明朝体漢字の表示が可能．

(2) CPUは80286(10MHz)を採用し，メモリは2Mバイトを標準実装．

(3) 3.5インチ・フロッピ・ディスク×1台，20/40Mバイト・ハード・ディスクを内蔵．

また同時に，以下のような2020シリーズの機能強化を行った．

(1) オフィス業務用ソフトをOFIS/CHART2，OFIS/FORM2として機能強化．

(2) 言語ライブラリとして各種イメージ・ファイルの入出力，画面表示，印刷を可能とした．

(3) 各種ネットワークとの接続機能．

【価格】20Mバイト・モデル：69万円，40Mバイト・モデル：78万円

㈱日立製作所　情報事業部　コンピュータ事業部
☎03(763)2411
〒140 東京都品川区南大井6-27-18

X-Window端末

クボタコンピュータは，米国Visual社製のX-Window端末X-19を発売した．これは14インチCRTモデルであるX640の後継機種で，19インチCRTディスプレイを採用したほか表示スピードの高速化を行ったもの．おもな特徴はつぎのとおり．

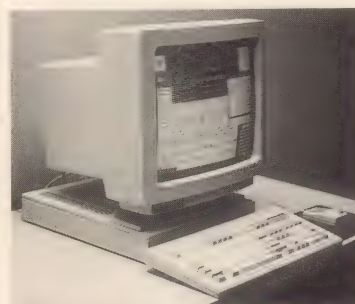
(1) ノン・インタレース式19インチCRTディスプレイを採用．

(2) CPU(MC68000)のクロックを16.6MHzに向上し，表示スピードを従来比2倍に高速化．

(3) IBM PC型とDEC VT220型の2種類のキーボードが選択可能．

(4) フォント・キャッシング機構により，あらゆるフォントを高速に利用することができる．

(5) ソフトウェアはX-Window System V11 R3.0，プロトコルはTCP/IP，Telnet，SLIP，NFS，ポートはIEEE 802.3，RS-232-C/422/423．

【価格】70万円(2Mバイト・メモリ，マウス，キーボード，ソフトウェア)

クボタコンピュータ㈱
☎03(225)0931
〒160 東京都新宿区新宿2-8-8

ハード・ディスク装置各種

●データ共有型ハード・ディスク

ランドコンピュータは，データ共有型ハード・ディスク・ユニットLHDU-40Cの上位機種としてLHDU-360を発売する．20/40/130/180 Mバイトの各カートリッジを2個まで装着することにより，最大360 Mバイトの記憶容量をもつ．

(1) 背面のマルチポートに4台までのコンピュータを接続して使用可能．さらに拡張アダプタを付加することにより最大8台までのパソコンでデータを共有することができる．

(2) 130/180 Mバイト・カートリッジには，電源が切れた場合に自動的にヘッドを退避するオート・リトラクト機能を搭載．

(3) 装置自体でのコピー機能や，パソコンからの電源ONを感知して自動的に電源を投入する機能をもつ．

(4) オプションのICカード・リーダを装備することにより，使用者の限定やデータ・チェックが可能．

【価格】LHDU-360(本体)：33万円，LHDC-180(カートリッジ180 MB)：60万円，LHDC-130(同130 MB)：45万円，LHDC-40C(同40 MB)：24万8,000円，LHDC-20C(同20 MB)：12万8,000円，SCSIインターフェース：9万8,000円

㈱ランドコンピュータ
☎06(304)8424/03(816)2671
〒532 大阪市淀川区西中島7-4-17

● SCSIハード・ディスク

ニューテックは，NEC PC-9801およびエプソンPCシリーズ用SCSIインターフェースを採用したハード・ディスク・システムNT100-55(100 Mバイト，平均シーク25ms)，NT40-55(40 Mバイト，同40 ms)を発売した．付属のインターフェース・ボードは，日本語MS-DOS ver 3.3または日本語MS OS/2 ver 1.0でサポートされているSCSIインターフェース・ボードとして使用可能．SCSIコントローラはWD33C93(ウェスタン,デジタル)を使用した．4台のドライブがデイジ・チェーンで接続可能で，1台のドライブを最大16パーティションで管理でき，パーティション容量は1 M～128 Mバイト(NT100-55では103.8 M)に設定可能．DMA転送のウェイトは無挿入．

【価格】NT100-55：25万8,000円，NT40-55：13万8,000円

㈱ニューテック　☎03(813)3891
〒113 東京都文京区湯島1-3-6
お茶の水Uビル

●大容量ハード・ディスク

キャラベルデータシステムは，光磁気ディスク・タイプを含むSCSI対応ハード・ディスクを発売する．容量は40 M～600 Mバイトまでの7タイプで，平均アクセス・タイムは16 ms．SCSIインターフェース・ボードPC-98M20は純正のPC-9801-55の上位コンパチブルで別売．

【価格】CA-6016SC(600 MB)：94万8,000円，-3016SC(300 MB)：58万8,000円，-6080MO(600 MB光磁気ディスク)：48万円，-2020SC(200 MB)：37万8,000円，-1025SC(100 MB)：21万8,000円，-0818SC(80 MB)：18万8,000円，-0428SC(40 MB)：12万8,000円，PC-98M20：3万8,000円

㈱キャラベルデータシステム
☎03(498)5370
〒150 東京都渋谷区渋谷4-3-17-606

● PC-9801用

ロジテックは，PC-9801シリーズ用の80 Mバイト・ハード・ディスク装置LHD-38VSを発売した．おもな特徴はつぎのとおり．

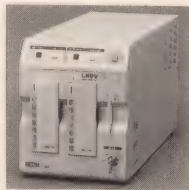
(1) FIFOメモリを応用したカスタムICを採用し，従来機種の約2.7倍のデータ転送を実現した．

(2) 40 Mバイト×2または80 Mバイト×1のドライブとして使用可能．平均アクセス・タイムは20ms．

(3) 80 Mバイト単一モードでもMS-DOS(ver 3.3以降)の立上げが可能．

(4) 80 Mバイト・モードでは16ビットFATに対応し，クラスタ・サイズを4 Kバイトとすることができる．

【価格】24万8,000円

㈱ロジテック　☎03(251)3271
〒101 東京都千代田区外神田2-15-2 新神田ビル

5相インテリジェント・ドライバ

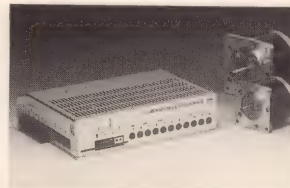
メレックは，パルス発振機能付きの5相インテリジェント・ドライバ・ユニットCD5500Sシリーズを発売した．おもな特徴は下記のとおり．

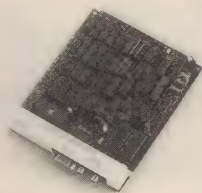
(1) シーケンサに直結が可能で，各種データはパネルのロータリ・スイッチから設定．

(2) モータとの結線が5本で行える．

(3) 最大消費電力を4段階に設定でき，モータ・ドライバの発熱を抑えることが可能．

(4) 機械原始点検出，原点復帰，CW/CCWリミット信号入力機能などをもつ．

【価格】8万9,000～10万2,000円

㈱メレック　☎0426(64)5382
〒193 東京都八王子市散田町5-8-20

PC-9801用ディジタル音声データ処理ボード

カノープス電子は，EIAJのディジタル・オーディオ・インターフェース規格に準拠した音声データ(CDやDATなどのデータ)をPC-9801に入出力するためのインターフェース・ボードDAIS-98を開発，販売を開始した．また，同社のPC-9801用DSPボードFLASH-16用の画像圧縮ライブラリ，また画像圧縮専用ボードFLASH-16 VIも発売すると発表した．

◈

DAIS-98の特徴はつぎのとおり．

(1) DMA転送+FIFOメモリ方式を採用し，Basicプログラムからでも簡単にデータの出し入れができる．

(2) サブコードのCビットおよびUビットを自由に読み書きできる．これにより，ディジタル・オーディオ機器のすべての信号のモニタおよび信号生成が行える．

(3) 簡単な信号生成およびモニタができるソフトが添付されている(全ソース公開)．

(4) RAMディスクとRAMディスク・ドライバがあれば，64Kバイトを越えるデータを扱える．

<おもな仕様>

入出力：各1チャネル(Z=75Ω，0.5 V_{P-P}，EIAJ CP-340)

サンプリング周波数：32.4 k/44.1 k/48 k/OPT

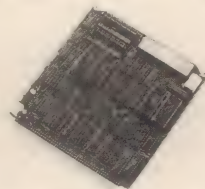
DMAインターフェース：PC-9801のDMAコントローラI/F内蔵

【価格】19万8,000円

◈

画像圧縮ライブラリは，フレーム・メモリやファイルに入っている画像データをコサイン変換/逆コサイン変換することにより，圧縮/伸長するもので，FLASH-16のDSP ZR34161用のものである．ただし，BasicやMS-C，MS-Fortranなどから，サブルーチンのように使える．

一方，FLASH-16 VIは，同社のDSPボードの従来機種FLASH-16を画像圧縮専用にカスタマイズしたものである．

【価格】画像圧縮ライブラリ：2万円，FLASH-16 IV：19万8,000円

カノープス電子(株)　☎078(411)5292
〒658 神戸市東灘区西岡本1-4-30

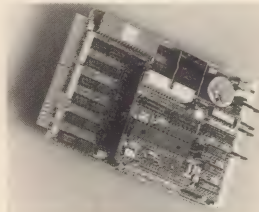
STD DOSボード用コプロセッサ・ボード

シスコンは，米Ziatech社が開発したSTD　DOSボード用コプロセッサ・ボードZT8832 I/Oプロセッサの販売を開始すると発表した．

STD DOSボードは，STDバス仕様の8088ボード・コンピュータにPC-DOSとBIOS(IBM社のライセンス付き)を搭載したものである．ZT8832ボードは，1個のCPUだけでは実行不可能な高速のI/O制御を分散して行うもので，以下の機能を内蔵する．① V40(8 MHz)，② 512 Kバイト ROM，③ 32 Kバイト2ポートRAM，④ SBXマルチモジュール用ソケット，⑤ 割込みコントローラ，⑥ 16ビット・タイマ・カウンタ，⑦ RS-232/485×2ポート，⑧ ウォッチドッグ・タイマ，⑨ 24ビット・パラレル・ポート，⑩ STD DOSドライバ．

また8832は，1ボード・コンピュータとしても使用できるほか，専用のラダー論理OSを搭載することもできる．

【価格】8832：14万1,000円，8832+ラダー論理OS：35万1,000円

(株)シスコン　☎03(863)7688
〒101 東京都千代田区岩本町1-3-7

PC-9801用磁気テープ装置

昭和情報機器は，PC-9801用磁気テープ装置PCMTの販売を開始した．特徴はつぎのとおり．

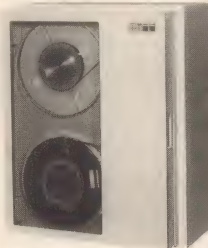
(1) 汎用機，オフコンなどとのデータ互換を考えたオープン・リール型を採用した．

(2) 記録密度は，800 BPI(NRZI)，1600 BPI(PE)に対応する．テープ速度は45 IPS．また，2400フィートまでのリールを装着可能．

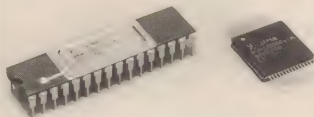
(3) 1台のコントローラで最大4台のデッキを接続できる．

(4) I/O制御ソフトは，Cの関数のかたちでサポートされている．

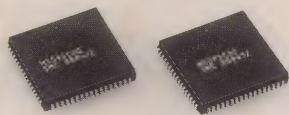
動作環境としてMS-DOS(ver 3.1以上)，MS-Cが必要となる．

【価格】コントローラ+ケーブル+ソフト：20万円，デッキ：140万円

昭和情報機器(株)　☎03(262)3061
〒102 東京都千代田区飯田橋2-13-7

256 Kビット Bi-CMOS ECL RAM

富士通は，64 K×4ビット構成のBi-CMOS ECL RAM MBM10C-504/100C504を開発，販売を開始すると発表した．それぞれ10 K/100 Kシリーズに対応する．アクセス時間は15 ns(最大)，消費電力は10C504が0.93 Wで100C504が0.65 W(標準)．32ピンDIPと28ピンSOP．

【サンプル価格】1万5,000円から

富士通(株)　☎03(216)3211
〒100 東京都千代田区丸の内1-6-1

SCSIコントローラとARCNETコントローラ

東洋マイクロシステムズは，米スタンダード・マイクロシステムズ社の開発したSCSIコントローラMSD95C00と，ARCNETコントローラCOM90C65/90C62の販売を開始すると発表した．

◈

SCSIコントローラMSD95C00は，ANSI X3T9.2に準拠したもので，以下の特徴をもつ．

(1) SCSI，DMAおよびCPUバスのそれぞれ独立した三つのチャネルをもち，DMA中でもリング・バッファの誤り訂正などができる．

(2) 8ビットの汎用ポートも内蔵しており，周辺チップの制御にも利用できる．

(3) コマンドの自動実行機能をもっており，バス・フリー・フェーズ，アービトレーション・フェーズ，セレクション・フェーズをレジスタの設定により，CPUの介在なしに自動実行できる．リトライ機能ももつ．

(4) 24ビットの転送カウンタ，12バイトのデータ・バッファを内蔵する．

(5) 最大転送速度は，非同期時で3 Mbps，同期時で5 Mbpsである．

CMOSプロセスで，パッケージは68ピンPLCC.

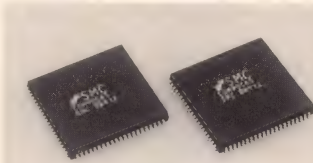
【サンプル価格】3,500円

◈

ARCNETコントローラCOM90C62は，同社従来機種COM90C26とエンコーダ/デコーダCOM90C32，クロック発振回路，パワーオン・リセット回路を1チップ化したもので，一方COM90C65は，さらにアドレス・デコーダなどの周辺回路も内蔵したものである．

なおARCNETは，トークン・パッシング・バス方式のLANで，① 転送速度2.5 Mbps，② 最大ノード数255，③ パケット・データ・サイズ最大508バイト，④ 最大ケーブル長6.4 km，⑤ トークン紛失時のネットワーク自動再構築機能，などの特徴をもつ．両チップには，これらの特徴をもったARCNETプロトコルが内蔵されている．

パッケージは，COM90C62が44ピンPLCCまたは40ピンDIPで，COM90C65が84ピンPLCC.

【サンプル価格】COM90C62(DIP)：7,300円，COM90C65：8,300円

東洋マイクロシステムズ㈱
☎03(423)6651
〒107 東京都港区赤坂4-9-17

ISDN用IC 2品種

● Q/Sチャネル支援のトランシーバ

インテルジャパンは，ISDNトランシーバ29C53AAの販売を開始すると発表した．29C53AAは，従来機種29C53をアップグレードしたもので，CCITT(I.430)とANSIにより標準化された“Q”および“S”チャネルをサポートする4線式全二重ディジタル・トランシーバである．Q，Sチャネルは，2ウェイの物理層で，ISDNの端末と交換機間にメンテナンス・データを送るためのもの．29C53AAの特徴はつぎのとおり．

(1) スペア・ビット・プロセッサ(29C53AAのハードウェア・ユニット)を使うことにより，ISDN端末(TE)とネットワーク・ターミネータ(NT)間の物理層メンテナンスを行うためのマルチフレーミング/ロー・スピード・チャネルをサポートする．

(2) マルチフレーミングを通してスペア・ビットを結合させることで，Qチャネルが形成される．TEからNTへの信号はQチャネルを介して，逆方向はSチャネルを介して送られる．

(3) ISDNのSループの両端で使用できる．

CMOSで，28ピンDIP/26ピンLCC.

【価格】2,450円(1000個時単価)
インテルジャパン㈱ ☎029747-8511
〒300-26 茨城県つくば市東光台5-6

◈

●電話用PCMコーディック/フィルタ

NSジャパンは，ISDN電話用のPCMコーディック/フィルタTP-3075/3076(COMBO II)の販売を開始すると発表した．これは，送信用バンドパス・フィルタと受信用ローパス・フィルタ，PCMエンコーダ/デコーダを1チップに収めたもので，以下の特徴をもつ．

(1) μ則/A則機能の切替えが可能．

(2) 300 Ωの駆動能力をもつ．

(3) 送/受信が独立して0.1 dBステップで25.4 dBまで可変できる．

(4) アナログ/ディジタル・ループバック機能をもつ．

(5) CCITT/LSSGR規格に準拠．

3075は28ピンDIPまたはPLCCで，3076は20ピンDIPである．

【サンプル価格】各2,200円
ナショナルセミコンダクタージャパン㈱ ☎03(299)7000
〒160 東京都新宿区西新宿4-15

ディジタル・ポテンショメータIC

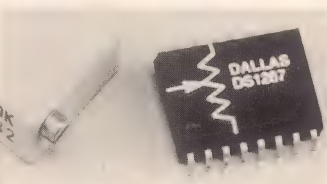
マイクロテックは，米ダラス・セミコンダクタ社のディジタル・ポテンショメータIC DS1267Sの販売を開始すると発表した．これは，抵抗値をソフトウェアで設定できる16ピンSOPのモノリシックICで，以下の特徴をもつ．

(1) 256個のタップをもった抵抗器を2個内蔵し，それぞれを独立して使うことも，スタック接続して512個のタップをもつ抵抗としても使える．

(2) 3本の信号線を接続することにより，縦続接続することもできる．

(3) 設定値は内蔵レジスタにデータを転送することにより行う．

10 kΩ，50 kΩ，100 kΩの3種類が用意されている．

【価格】690円(100個時単価)

マイクロテック㈱ ☎03(371)1811
〒160 東京都新宿区西新宿7-9-17

●機能強化版 iRMX および 80960 用リアルタイム OS カーネル ── iRMX I.8, iRMX II.4, iRMX960

インテルジャパンは，16 ビット・マイクロコンピュータ用 OS 2 種類 iRMX I.8, iRMX II.4 および i80960 32 ビット・マイクロプロセッサ用リアルタイム OS カーネル iRMX960 の販売を開始した．それぞれの特徴は，以下のとおりである．

▶ iRMX I.8

(1) ラウンド・ロビン方式のスケジューラが追加された．

(2) 端末サポート・コードを変更することにより，シリアル I/O のスループットを 200 %以上向上させている．

▶ iRMX II.4

(1) マルチバスIIシステム・アーキテクチャ(MSA)を通じてシステム・ブートがサポートされる．

(2) iRMX I.8 と同様，端末サポート・コードを改善することにより，シリアル I／O のスループットを向上させている．

(3) 総合グラフィック・ソフトウェアが付加された．

▶ iRMX960 リアルタイム・カーネル

(1) カーネル，リアルタイム・サービスを実現するコア・モジュール，コア拡張用のオプショナル・モジュール，および周辺デバイス管理ソフトウェアで構成される．

(2) 80386 用 iRMX I カーネルとの互換性があり，アプリケーション・ソフトのポータビリティが得られている．

【価格】iRMX I.8 が 100 万円，iRMX II.4 が 150 万円，iRMX960 が 60 万円

インテルジャパン㈱

☎ 029747-8511

〒300-26 茨城県つくば市東光台 5-6

◈

●パーソナル CAD 関連シリーズ・ソフトウェア 4 品種 ── OrCAD/PCB, OrCAD/PLD, OrCAD/MOD, ALS-VIEW

ソーテックは，パーソナル CAD ソフト OrCAD の関連商品として，OrCAD/PCB, OrCAD/PLD, OrCAD/MOD, ALS-VIEW の販売を開始した．それぞれの機能は，以下のとおり．

▶ OrCAD/PCB(59 万 8,000 円)… PCB 基板レイアウト・ソフト．① 約 81 cm×81 cm, 2000 ネット，1850 パッズ，IC 130 個までのプリント基板を扱える，② オート・ルーティング，マニュアル・ルーティングの機能を備える，③ 約 150 種類のデバイスの外形のライブラリを備える，などがおもな特徴．

▶ OrCAD/PLD(14 万 8,000 円)… プログラマブル論理デバイス設計ソフト．① 論理の記述方法として，2 進値，真偽表，回路図の中から選択できる，② いくつかのブール代数式をまとめて一つのインデックス方程式として定義できる，③ ライブラリとして標準的な PLD を用意している，などがおもな特徴．

▶ OrCAD/MOD(14 万 8,000 円)… PLD もシミュレーションできるように OrCAD/VST の機能を拡張するソフト．① 300 以上のタイミング・モデルをライブラリとして備える，② 標準モジュールと PLD モジュールを組み合わせてタイミングをテストできる，などがおもな特徴．

▶ ALS-VIEW(39 万 8,000 円)… ガーバー・データの変更をパソコン上で行えるようにするソフト．

㈱ソーテック　☎ 045(662)0688

〒231 横浜市中区太田町 4-55

◈

● LAN 対応 VT282 相当端末エミュレータ ── J-TOOL

アライド・テレシス㈱は，LAN (TCP/IP)に対応する PC-9801 シリーズ(および互換機)用 VT282 相当端末エミュレータの販売を開始した．このソフトにより，LAN 経由でパソコンを VT 端末として使用できる．おもな特徴は，以下のとおりである．

(1) DEC 漢字のほか，シフト JIS, 新 JIS, 旧 JIS, EUC 漢字コードをサポートする．

(2) VT 端末エミュレーション付き telnet(VTN)，同リモート login (VRL)，漢字変換機能付き ftp (JFTP)，漢字ファイル表示/変換ユーティリティのソフトウェアがサポートされる．

【価格】 2 万円

アライド・テレシス㈱

☎ 03(443)5640

〒141 東京都品川区東五反田 4-5-2

◈

●富士通 EWS S ファミリ用マイコン・クロス開発支援システム ── Exella-XP

三岩商事は，ファームウェアシステム㈱が開発したマイコン・クロス開発支援システム Exella-XP(富士通 EWS S ファミリ版)の販売を開始した．これは，クロス C コンパイラ，クロスアセンブラ，ICE デバッガの専用ツールを MPU ごとに用意したもので，MPU としては，68020, 68000, 8086, H16, Z80 などをターゲットとしている．C 言語の仕様は，Unix の PCC に準拠し，アセンブリ言語仕様は，チップ・メーカに準拠している．ワークステーションから LAN で接続された複数のパソコンを開発用の端末として利用できるとともに，パソコン側に ICE を接続することにより，ICE 分散デバッグが可能となる点が特徴．

【価格】 S-3 用が 120 万円，S-4 用が 180 万円

三岩商事㈱　☎ 03(407)2181

〒150 東京都渋谷区渋谷 3-15-6

◈

●パソコン間でネットワークを実現するための通信ソフト ── PC ブリッジ

インターコムは，㈱リギーコーポレーションと共同で，パソコン・ネットワーク・システム構築用の通信ソフト PC ブリッジを開発し，販売を開始した．これは，MNP モデムを使い，電話回線を通してパソコン同士を接続し，データの交換や共有化を可能にするもので，PC-9801 シリーズおよびエプソン PC-286 シリーズのパソコンに対応する．おもな特徴は，以下のとおりである．

(1) MS-DOS 形式のファイル(文書/データ)を送受信できる．

(2) ダイヤリング，ファイル送信，ファイル受信を自動的に運転できる．

(3) 相手のパソコンを MS-DOS コマンドで直接操作できるリモート DOS 機能を備える．

(4) パスワードや相手電話番号などの自動識別でデータの機密を保持するデータ・セキュリティ機能を備える．

(5) そのほか，複数の相手パソコンへの同報通信機能や送信/受信の自動切替え機能などを備える．

【価格】 5 万 8,000 円

㈱インターソフト　☎ 03(293)3338

〒101 東京都千代田区神田駿河台 2-9

◈

●プリント板設計・製造システム
── ICAD/PCB4(V15)

富士通は，プリント板CIMの中核となるプリント板設計・製造システムICAD/PCB(V15)の販売を開始した．FACOM Mシリーズ(適用OSはOSIV/MSP，OSIV/FSP)で稼働するもので，以下のモジュールで構成される．

●基本モジュール…画面表示やデータベース管理などを行う．
●アナログ・パターン設計モジュール…部品や配線パターンの入力・編集を会話的に行う．
●回路設計モジュール…回路図の入力や部品の割付けを会話的に行う．
●ライブラリ・モジュール…設計基準データ，部品データなどプリント板設計で必要な情報を作成する．
●図面入力支援モジュール…回路図面やパターン設計図面を図面入力装置で入力する際に必要なモジュール．
●アナログ製造データ変換モジュール…アナログ設計データをNCデータ出力の前提となる製造インターフェース・ファイルに出力する．
●アートワーク・モジュール…フォト・プロッタ用データを出力する．
●ドリル・モジュール…プリント板穴あけデータを出力する．
●インサータ・モジュール…部品実装用NCデータを出力する．

【価格】標準的なシステム構成(基本モジュール＋アナログ・パターン設計モジュール＋ライブラリ・モジュール)で，110万円/月から
富士通㈱　☎03(216)3211
〒100 東京都千代田区丸の内1-6-1

◈

●RS-232-Cコミュニケーション・アナライザ ── RCA98

アドテックシステムサイエンスは，RS-232-Cシリアル通信モニタRCA98の販売を開始した．これは，PC-9801シリーズのシリアル通信部分のデバッグを行うためのプログラムであり，通信状態をCRT画面に表示し，不具合などを確認するものである．疑似的にシリアル・データを作成し送信することもできるので，シリアル伝送装置のデータ/ライン検査に応用できる．そのほか，①任意の送信データを登録し，再送信するための入力キーの保存機能，②DSR, CTS, CI, DCDの各ラインの状態を表示し，DTR, RTSラインを制御する機能，③送受信したデータ数やエラーの発生状況をリアルタイムに表示する機能，などを備える．

【価格】1万9,800円
㈱アドテックシステムサイエンス
☎045(331)7575
〒240 横浜市保土ヶ谷区天王寺1-16-6

◈

●英日双方向電子辞書システム
── X-DIC

スピリットは，パソコン用翻訳支援システムとして，英日双方向電子辞書システムX-DICの販売を開始した．おもな機能は，以下のとおり．

(1) 基本辞書として，3万語の英和辞書および3万語の和英辞書が内蔵され，またそのほかに，ユーザ辞書として1Mバイトにつき3万語を登録できる．
(2) ウィンドウを用いた編集機能を備え，スクリーン分割で，英文と日本文を同時に操作できる．
(3) ファイル化された翻訳前のファイルに対して，一括辞書引きができ，翻訳ファイル作成までの無人化を可能にしている．
(4) MS-DOSテキスト形式で，他のソフトとのデータの互換性がある．
(5) 変換速度は，英日翻訳で7,000語/時間，日英翻訳で2,500語/時間である．

適応機種は，PC-9801シリーズ，FM Rシリーズ，松下Mシリーズ，日立B16シリーズ，AXパソコンである(MS-DOS ver 2.11以上)．

【価格】9万8,000円
スピリット㈱　☎03(234)8041
〒102 東京都千代田区三番町7

◈

●Turbo C ver 2.0 AXパソコン版

動作環境は，MS-DOS ver 3.2以上(メモリ640Kバイト以上)．既存のIBM PC用あるいはPC-9801用のアプリケーションをAXパソコンに移行させることができる．

【価格】2万9,800円
㈱ボーランドジャパン
☎03(486)1400
〒107 東京都港区南青山7-8-1

バージョンアップ製品

●Lattice C/DOS ver 4.1

Lattice C ver J4.0をバージョンアップするとともに，MS-DOS専用版として製品化したもの．①デバッガがソース・ウィンドウにソース・アセンブリ混合リスト，アセンブリ・リストの表示を可能にしている，②グローバル・オプティマイザの機能を強化し，不要なループの最適化や未参照ローカル変数への代入の消去を行う，③huge，volatileの二つのキーワードを新たに追加した，④その場でコード展開して実行できるビルトイン関数をサポートした，などがおもなバージョンアップ内容．

対応機種は，PC-9801シリーズ，FM Rシリーズ，FM-16β，Panacom Mシリーズ，J-3100シリーズ，PS/55シリーズである．

【価格】9万8,000円
㈱ライフボート　☎03(293)4711
〒101 東京都千代田区神田錦町3-6

◈

●ゲートアレイ/スタンダード・セル開発用ツール ── VISTASL III

PC-9801シリーズ上で実行可能なゲートアレイ/スタンダード・セル開発用ソフトウェア・ツールで，VISTASL IIの後継ソフトウェアである．回路図の作成，テスト・パターンの作成，シミュレーションおよびその結果解析を行う．①OSとしてMS OS/2を採用した，②最大シミュレーション・ゲート数を20Kゲートに拡大した，③シミュレーションの処理速度を従来の3倍以上にした，などがおもなバージョンアップ内容．

【価格】50万円
日本電気㈱　☎03(798)6511(広報)
〒108 東京都港区芝5-33-1

◈

●C-ISAM ver 2

おもなバージョンアップ内容は，以下のとおり．

(1) ISAMファイル仕様は，dBASE III，dBASE III plusおよびdBASE IVに対応する．
(2) 従来のデータ・ファイル(DBF)，インデックス・ファイル(NDX)，メモ・ファイル(DBT)に加えて，1ファイルで最大47個のインデックス・ファイルを扱うことができるマルチインデックス・ファイル(MDX)を新たにサポートする．

動作環境は，MS-DOS ver 3.1以上，Lattice Cコンパイラver 4.

【価格】2万9,800円
㈱ライフボート　☎03(293)4717
〒101 東京都千代田区神田錦町3-6

7月号特集「Cによるマルチタスク OS 入門」について

■入門とするには難しい

▷今月の特集は，なかなか読みごたえもあり，マルチタスク OS の整理にはもってこいである．ただ，やはり入門とするには少し無理があるかもしれない(記事のばらつきがないが，レベルが高い)．できれば，PC-98 や J-3100 あたりのソースを配布してほしいと思うのだが．(大阪府守口市・中井一人・30・㈱松下ソフトリサーチ第1開発室)

▷今回の特集は少しむずかしかった．とくにアドレス変換論理のところは，理解に苦しんだ．筆者の方は図を工夫してわかりやすくなるように書いているのがわかるので，自分の理解力の不足がわかった．これとは別に各記事の筆者について，どんな人で今どんなことをしている人か少し解説をつけてもらうと，さらに記事に対する見方が変わるのではないだろうか．(埼玉県川越市・孝山裕秀・39・川研ファインケミカル㈱施設部施設課)

▷今回の特集は，非常に興味をもって読んだ．業務にすぐ役立つわけではないが，マルチタスク OS が何をやっているのかがおぼろげながら見えてきたように感じた．(新潟市・山崎学・38・新潟通信機㈱ 情報機器課)

▷ CPU の記述には，適宜 80286 や 80386 との対応箇所と相違箇所などを加えてほしい．(岐阜県真正町・森口博文・28・岐阜工業高等専門学校)

➔一般記事などについて

▷意外な記事(I/F 誌としては)でよかったのが，「Mac のファイル・システム」，「Prolog」，「PDS」，「コピー・プロテクト」など．ハードがりがりの記事もいいけれど，ソフトのこんな色っぽい記事もいい．最近は，I/F 誌もけっこう硬軟の使い分けがうまくなったと思う．2，3 年前はむずかしく見えて「つん読」だったのに，今は月の 25 日が待ち遠しい．(京都市・椎野倫人・32・滋賀医大)

▷割込みについての今回の記事は，すっきりしていたのでよかった．MS-C ver 5.1 でも Turbo C でも Power C でも，同じことができるので，いろいろ応用させていただきたい．(金沢市・森下公博・31・金沢市立工業高校)

▷(「割込みとシステム・コールの問題とその解決法」について) MS-C や Turbo C の interrupt 関数は，本来はソフトウェア割込みの記述用にあるのではないか．なぜなら interrupt 関数に引数が指定でき，この引数で割込み関数に入った時点での各レジスタが参照/変更できるからである．ROM BIOS や INT 21H は，入力/出力ともレジスタを使用しているから，既存の割込みに割り込んだり，別個に記述したりが簡単にできるわけである．したがって，これを無理やりハードウェア割込みの記述に使用するといろいろな制限が出てくるのは当然ではないだろうか(ただし，岡村氏の手法を否定しているわけではない)．(広島市・中島信行・35・㈱ Unix)

9月の催し物/おしらせ

◈ホーム・オートメーション'89

〔開催日〕9月6日(水)～9日(土)
〔場所〕東京国際見本市会場(東京都中央区晴海)
〔内容〕'89 住宅設備展に併設されるもので，HA 技術をコアとしたホーム・アメニティをつくり出すためのシステムや機器が展示される．
〔入場料〕無料
〔問合せ先〕日本能率協会住宅設備展シンポジウム事務局 ▶〒105 東京都港区芝公園 3-1-22 ☎03(434)1391

◈ソフトウェア・ショウ'89

〔開催日〕9月27日(水)～29日(金)
〔場所〕サンシャインシティ・ワールド・インポート・マート(東京都豊島区池袋)
〔入場料〕無料
〔問合せ先〕㈶ソフトウェア情報センター内ソフトウェア・ショウ'89 事務局 ▶〒105 東京都港区虎ノ門 5-1-4 東都ビル 5F ☎03(437)3394

◈ IC カード＆光カード展

〔開催日〕9月27日(水)～30日(土)
〔場所〕東京流通センター(東京都大田区平和島)
〔入場料〕1,000円
〔問合せ先〕日本工業新聞社事業部 ▶〒101 東京都千代田区神田神保町 1-28-5 ☎03(292)3561

◈セミナー「生産現場での誤動作原因の究明と実践ノイズ対策」

〔開催日〕東京：9月25日(月)，大阪：10月3日(火)
〔場所〕東京：国立教育会館(千代田区霞が関)，大阪：大阪科学技術センター(西区靭本町)
〔内容〕㈱電研精機研究所ノイズトラブル相談室 平田源二氏が，ノイズ対策を実施するうえでの基礎知識，誤動作原因の究明，設備機器のトラブルと対応，工場内でのノイズ対策，などについての講演を行う．
〔受講料〕2万8,000円
〔問合せ先〕㈱経営コンサル ▶〒105 東京都港区新橋 2-10-9 ☎03(501)6811

◈ MICROWAVE USA '89 展/技術セミナー

〔開催日〕9月26日(火)～29日(金)
〔場所〕サンシャインシティ・ワールド・インポート・マート 7F U.S.トレード・センター展示場
〔内容〕マイクロウェーブ製品とその技術の展示および技術セミナーの開催
〔入場料〕無料
〔問合せ先〕U.S.トレード・センター ▶〒170 東京都豊島区東池袋 3-1-3 ワールド・インポート・マート 7F ☎03(987)2441

◈日本システム，本社を移転

日本システム㈱は，8月15日付で本社を下記に移転した．
新住所：〒162 東京都新宿区住吉町 8-12 ☎03(226)3600

◈ダイヤセミコンシステムズ，営業本部事務所を移転

ダイヤセミコンシステムズ㈱は，7月31日付で，営業，技術，業務部門を下記に移転した．
新住所：〒154 東京都世田谷区新町 1-23-9 フラワーヒル新町東館 1F ☎03(439)1600(マイクロコンピュータ営業本部)，03(439)2700(ストラテジックマーケティング本部)，03(439)4141(三菱半導体営業本部)

編集デスクから

▶**特集**…編集者の率直な感覚として，OS-9/68K を2号にわたって特集するというのは，少し冒険かなとも思いました．OS-9/68K が，基本的にそれほどポピュラーではないからです．もちろん冒険しようと思って企画したわけではなく，半年ぐらい前から OS-9/68K の特集を組んでほしいという読者からの葉書が増えだしたということがあったからです．おそらく X68000 が OS-9/68K をサポートするようになったことも一つの大きな要因でしょう．そういう意味で，「ホビイスト向けのマシン」だといわれている X68000 に関する記事も意識的に取り入れてみました．一方，すでに先月号に対する読者からの声も届いてきていますが，OS-9/68K の記事に興味をもたれた読者の多くが，制御関係の開発に携わっているという点も，うなずけることです．

▶もっともそれだけではなく，水面下では(あるいは今月号が書店に並ぶころには表だっているかもしれませんが)，OS-9/68K を取り巻く状況に大きな動きがあることも確かです．OS-9/68K がもっとメジャーになることを期待して，いつか再度特集として取り上げたいと考えています．

▶なお，OS-9/68K の制御システムへの応用や PC-9801 用 OS-9/68K ボードの詳細に関する記事も紹介する予定でしたが，編集部の都合により掲載できませんでした．

▶**連載**…今月号から「グラフ理論」の連載を開始しました．実用的というタイトルにもあるように，理論と現場の技術との接点を求めてのことです．グラフ理論は，モノとモノの関係だけに注目するという単純な始まりだけに，状態遷移やネットワークなど応用範囲は，かなり広いと思われます．

でばっぐ

8月号

● **MS-DOS のメモリ管理の…**

p.244 図14 中，上から2個めの図の中の「バッチ・ファイル管理ブロック」の MCB が示す PSP セグメント・アドレスは，PSPb の誤り．p.246 右段↑14，p.248 右段↓1，↓2，↓5にある〈再帰呼出し〉は，〈再入〉の誤り．

和書新刊 (価格には，消費税が含まれていない場合があります)

●**計算機言語**(経営情報学講座 11) 下條哲司ほか監，A5判，150頁，2,369円，オーム社

●**コンパイラ理論**(コンピュータ数学シリーズ 10)　大山口通夫著，A5判，180頁，2,000円，コロナ社

●**シソーラス構築法**　内藤ほか訳，B5判，208頁，2,781円，丸善

●**通信制御インターフェース**(原典 CTRON 大系 5)　坂村健監，B5判，492頁，15,450円，オーム社

●**通信制御インターフェース ISDN ユーザ制御編**(原典 CTRON 大系 7)　坂村健監，B5判，462頁，13,905円，オーム社

●**実行制御インターフェース保守運転管理インターフェース**(原典 CTRON 大系 8)　坂村健監，B5判，288頁，8,755円，オーム社

●**情報ネットワーク**(ISDN 入門シリーズ 1)　秋山稔著，A5判，230頁，2,400円，コロナ社

●**信号処理工学**　虫明康人監，A5判，164頁，2,884円，オーム社

●**ソフトウェア技術者のためのハードウェアの基礎**　小野瀬一志著，A5判，322頁，3,399円，オーム社

●**ディジタル信号処理のポイント**　石田義久・鎌田弘之著，A5判，250頁，2,800円，産業図書

●**並列処理機構**　高橋義造編，A5判，280頁，4,635円，丸善

●**ACM チューリング賞講演集**　ACM Press 発行，菊判，600頁，9,270円，共立出版

●**TEX(テフ)入門**　大野義夫編，A5判，256頁，2,750円，共立出版

●**ニューロコンピュータへの発想**　甘利俊一監，B6判，180頁，1,550円，共立出版

●**機械翻訳**(ニューメディア技術シリーズ)　電子情報通信学会編著，A5判，222頁，2,781円，オーム社

●**シグナルプロセッサのソフトウェア**　石井六哉・小野定康著，A5判，180頁，1,800円，コロナ社

●**VM**(OS シリーズ 11)　岡崎世雄・全先実著，A5判，232頁，2,730円，共立出版

●**プログラムの構造と実行(下)**　エーベルソン著/元吉文男訳，A5判，320頁，3,500円，マグロウヒル出版

洋書新刊 (洋書価格は丸善調べで，すべて予価)

● ***An Introduction to Neural Computing***　I.Aleksander，H. Morton 著，280頁，5,760円，Kogan Page，英国

● ***Analysis of Cache Performance for Operating Systems and Multiprogramming***　A. Agarwal 著，224頁，12,000円，Kluwer Academic，オランダ

● ***Introduction to occam 2 on the Transputer***　G.R.Brookes，A.J.Stewart 著，112頁，2,700円，Macmillan，英国

● ***occam 2***　J.Galletly 著，224頁，3,720円，Pitman，英国

● ***Transputer Application***　G. Harp 編，264頁，6,780円，Pitman，英国

● ***VMEbus User's Handbook***　S.Heath 著，320頁，11,900円，Heinemann Professional，英国

● ***Prolog versus You : An introduction to logic programming***　A.L.Johansson ほか著，297頁，6,380円，Springer，西独

● ***Supercomputing Systems : Architectures, design, and performance***　S.Kartashev ほか編，704頁，11,500円，Van Nostrand，米国

● ***Why Prolog ? Justifying Logic Programming for Practical Applications***　G.L.Lazarev 著，224頁，7,100円，Prentice-Hall，米国

● ***Computer Communication Technologies for the 90's***　J. Raviv 編，606頁，24,000円，North-Holland，オランダ

● ***Digital Signal Processing with the TMS320C25***　R.Chassaing，D.W.Horning 著，340頁，7,990円，Wiley，米国

● ***ISDN Design : A practical approach***　S.Hardwick 著，157頁，6,990円，Academic Pr.，米国

● ***An Introduction to Digital Signal Processing***　J.Karl 著，334頁，7,990円，Academic Pr.，米国

次号予告

10月号は9月25日発売です

特集 Cによるディジタル回路シミュレーション

C言語と論理回路／回路記述言語／論理回路ライブラリ

▷ディジタル回路の製作を行う場合，一般的にはひととおりの回路設計が終わった段階で実験回路を組み立て，オシロスコープやロジック・アナライザなどを用いて回路動作の確認を行い，不都合を修正して，最終的な回路を決定するというカット＆トライ手法がとられているのが実状である．単品ものならまだしも，ある程度の数量が見込まれる場合に，このような開発手法は時間と労力の無駄となる．

▷そこで従来からコンピュータによる回路動作のシミュレーション（模擬）が行われてきたが，汎用機やEWS上での使用はコスト的に見合わないということも多い．そこで次号の特集では，パーソナル・コンピュータ上で手軽に利用できるディジタル回路シミュレーションの実例を紹介する．

■シミュレーションの手法

▷紹介するシミュレータは，独自に定義した回路記述言語によって基本論理回路を定義し，それらをライブラリ化していくことで，より大きな応用回路の動作解析を行う構成となっている．C言語の関数とディジタル回路の階層構造との関連性に着目し，記述された回路をCのソース・プログラムに変換するコンパイル方式を採用している．

■シミュレーションの実際

▷後半は，組合せ回路，順序回路，応用回路のシミュレートを行って，回路設計のなかでのシミュレーションの実際について考えていく．作成したシミュレータは，74LSシリーズTTLのライブラリも含めて，ディスク・サービスを行う．

《別冊付録》 EMS活用ガイド

▷EMS対応のアプリケーションが増えてきている．そこで別冊付録では，EMSの情報をまとめる．EMS規格を整理し，EMSマネージャであるEMMの概要，市販EMS対応ボード，EMS利用手順などをコンパクトにまとめる．

編集余滴

●すこし前，部内で机の配置がえがあり，当編集部は窓際へ移りました．最近やっと慣れましたが，窓際になってよかったことがひとつ．ビルの谷間から，ほんのわずかですが空が見えるんです．あんな小さな空でも，校正の合間にながめると，目の保養になって能率があがります．しかし，その空に夏雲が見えだしたらイケマセン．心はアウトドアへ…で気もそぞろです．（Fra）

●カラヤンが死んだ．彼は，演奏会場の聴衆だけでなく，各種メディアの向こう側の聴衆をも強く意識した指揮者だった．メディア自体（ハード）にも興味をもちながら，メディアにのる自分の情報も自身で完全に制御した．彼の音楽以外のこうした趣味が演奏のなかで鼻につくこともあった．が，彼の実験的な試みが現代のメディア時代を招いたといっても間違いではないだろう．（矢）

●以前この欄で紹介したoftenとquite often の頻度の問題だが，ある米国人に尋ねたら，quite oftenのほうが頻度が高いということで，混乱してしまった．そこにまたまた混乱．supperはdinnerの後の軽い食事のことだと．それまでdinnerとsupperは代替的なものだと思っていたから，またしても耳を疑った．米国人は，豪州英語は違うんだとはいうものの….（yew）

●旧い言葉に“敵を知り己を知れば百戦危うからず”というのがあって，最近は敵の種類が多いせいか，敵を解説した文献が山ほどある．ここで敵とはシステム，すなわちOSであったりCPUであったり，あるいは言語やアルゴリズムであったりする．何千ページものマニュアル付きという敵さえいる．敵の情報は十分だ．さて，それでは己とは何か，どうやって知るのか．（富）

●眠れぬ夜が，三晩つづいた．夜ふかしはするが，寝つきのいい私にとって，快挙．でも，仕事や恋の悩みがあったりすればサマにもなるのに，理由が特に見つからない．“これが不眠症？”と半分喜んでいた翌日，目覚めたのはお昼すぎ．堂々の12時間睡眠を決行したのでした．いったいあの三晩はなんだったのだろう．今度は本当に悩んで眠れなくなりそうです．（Peko）

インターフェース　1989年9月号　第15巻　第9号（通巻第148号）

発行人 神戸一夫　編集人 金子俊夫
発行所 CQ出版株式会社 〒170 東京都豊島区巣鴨1-14-2 電話 03(947)6311～6315 振替 東京 0-10665
1989年9月1日発行（毎月25日発売） <定価は表四に表示してあります>

(社)日本ABC協会認証誌 ISSN 0387-9569 印刷 図印刷株式会社 美和印刷株式会社 製本 有限会社 経文社井上製本

3次元図形を扱うようになった

高速描画フル・カラーGDCの動向

このページは，CQ出版社発行の月刊・半導体情報誌データムの編集ページ「デバイス・トレンド」を再掲載したものです．データムはこれらの編集ページに加え，各社の新規データブックの紹介などを含め32ページ，そして半導体の個別詳細データ頁が毎号800ページほど掲載してあります．

小林純一

最近，半導体メーカ各社より多様な機能・構成をもつグラフィック・コントローラが市場に出ています．その多様化の背景にあるのは次の3点です．

(1) より高解像度のグラフィックスを，より高速に表示したいという市場の要請．

(2) 3次元画像，動画などの，より高度なマン-マシン・インターフェースへの要求．

(3) システムのトータル・コスト，スペースを軽減したいという機器メーカの希望．

まず1986年頃から直線や円弧の描画機能をもつCRTコントローラが市場に出てきました[8](表1)．これらのチップは，ウィンドウ・サポート，CAD，デスクトップ・パブリッシングなどのアプリケーションに求められる描画機能を，ホストCPUからコマンドで指示されることにより実行するという点で共通しています．これに対し，最近3次元グラフィック機能を有するチップの開発が盛んになってきました[2]．3次元グラフィックの特徴として次の3点があげられます[4]．

(1) ドット単位で色調がフル・カラー指向であること．これは曲面処理，陰影処理の結果，階調がリニアに表現できないと表示画像が不自然になることに起因します．

(2) 描画アルゴリズムが確立していないということ．現在，より自然な出力画像をより高速に得ようと，アルゴリズムの研究が続けられている段階であり，機能と速度のトレードオフにより，種々のアルゴリズムが提唱されています[3]．

(3) 描画以前の計算量が膨大であること[6]．これは，最近の32ビット・マイクロプロセッサを用いても単独では十分な処理能力が得られず，何らかの並列処理が不可欠となっています．

以上の3点により，従来の2次元図形を対象とするグラフィック・コントローラとは異なったホストCPUインターフェースをもつCRTCが必要となってきます．逆に3次元グラフィックを扱えるシステムは，2次元図形は容易に処理可能です．

◈ カラー・グラフィック・コントローラ G300

インモス社から発表されたカラー・グラフィック・コントローラG300[1]は，いままで述べた市場の要求をみたすものとして開発されました．

G300の特徴をまとめると次のようになります．

(1) VRAMとホストCPUとCRTを結合するための機能をすべて1チップに統合した．

(2) 1600万色(RGBそれぞれ8ビット)のフル・カラーを同時表示可能とした．

(3) 描画アルゴリズムに制限を加える要素を一切排除した．

それぞれのポイントについて詳しく見ていきます．

まずG300を用いたグラフィック・システムの構成例を図1に示します．ここで，ホストCPUからG300は512バイトのアドレス空間をもつSRAMとして見え，すべての初期化はこのポートへの読み書きによって行われます．またVRAMに対しては，シリアル転送用のクロ

発表年	1981	1984	1986			1987
型名	μPD7220	HD63484	82786	TMS34010	Am95C60	μPD72120
アーキテクチャ	ピクセル・パッキング	ピクセル・パッキング	ピクセル・パッキング	ピクセル・パッキング	プレーン	ピクセル・パッキング/プレーン
ウィンドウのサポート	―	ハードウェア	ハードウェア	BitBLT	BitBLT＋ハードウェア	BitBLT
文字描画	CG*	CG*	BitBLT	BitBLT	BitBLT	BitBLT
デュアル・ポートRAMのサポート	なし	なし	あり	あり	あり	あり

*CG：キャラクタ・ジェネレータ

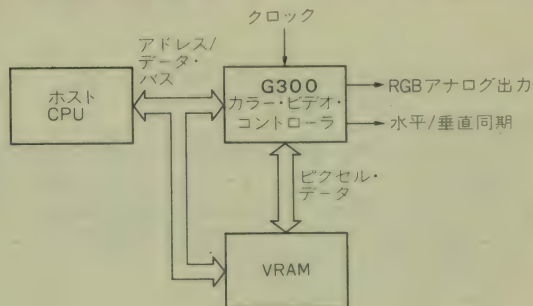
〈表1〉描画機能をもつCRTコントローラ

CQ出版社　〒170 東京都豊島区巣鴨1-14-2　☎03-947-6311　振替 東京0-10665

ックを供給し，VRAMの出力は32ビットのピクセル・ポートよりG300に入力します．

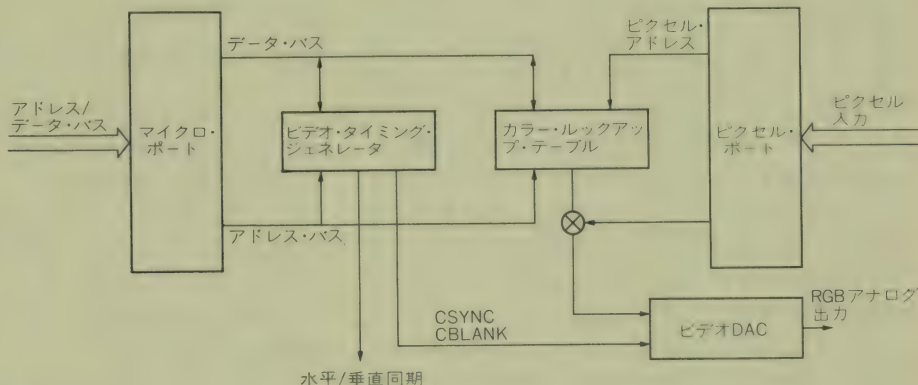
次にCRTには，アナログRGB信号，水平/垂直同期信号を供給しますが，内蔵のビデオ・タイミング・ジェネレータの各レジスタに適切な値を設定することにより，あらゆるラスタ・スキャン型ディスプレイとインターフェースにすることが可能です．また8ビットのビデオDACを3個内蔵しているため，外付けのビデオDACは不要です．

したがって，G300は，ビットマップからディスプレイまでのすべての制御を1チップにおさめたLSIであるといえます．G300の内部構造を図2に示します．

次に，3次元グラフィックスにおいて必要となるフル・カラー表示能力ですが，G300は70MHzの速度で1600万色の同時表示を行うことが可能です．この場合，VRAMはRGB×8ビット×(フレーム・サイズ)の構成をとることになります．またモードの切り替えにより，32ビットのピクセル・ポートからの入力を4バイトのピクセルと解釈し，8ビット→24ビットのカラー・パレットを経由して120MHzの256色表示を行うことも可能です．これらのモードの選択は，CPUから内部のモード・レジスタを書き換えることにより行います．

〈図1〉 G300を用いたグラフィック・システムの構成例

● **描画アルゴリズム**

最後に，G300の特徴のひとつであるアルゴリズムの

〈図2〉
G300内部ブロック図

RenderMan

RenderManとは，3次元モデリング・システムとレンダリング・システムとの間のインターフェース標準のことです．3次元モデリングでは，ある情景の中の物体の形状，位置関係，材質をデータや手続きとして表現します．またレンダリングでは，ある視点から見た情景の映像を立体的に表現します．

RenderManは具体的には，レンダラが映像を生成するために必要な，あらゆる情報を表現し得る約百個のサブルーチンの仕様という形で提供されます．さらに物体の再帰的な性質(例えば植物や雲，火災など)を表現するような記述言語も含まれています．

表現できる物体の特徴としては形，色，透明度，光の反射率などです．このうち形は各種の曲面関数が使えますし，色はたんなるRGBのほかにスペクトラムの複数の波長によっても表現できます．また視点側のカメラのレンズ特性を加味することも可能です．

以上の強力な表現能力を用いて，写真のように現実感のある画像が得られるわけです．

RenderManは，最新の画像処理に関する研究成果を取り込んで将来のグラフィック・システムの目標となるべく作成された標準です．そのため，これまでのグラフィック標準のように確定したころにはすでに陳腐化してしまっているということはないでしょう．

RenderManバージョン3.0の資料は米国Pixer社より入手することができます．

独立性について述べます．

G300では描画機能はすべてホストCPUに任せて，VRAMからCRTへの表示機能を担当する設計となっています．この理由は，上で述べたように3次元グラフィックスのアルゴリズムが適用分野によって異なり，また各分野での最適なアルゴリズムが進化しつつあるという現状によるものです．

ホストCPUとして，高度の並列処理を目的として設計されたトランスピュータT800（インモス社）を使う場合を考えます．この場合，T800とG300を組み合わせることにより，例えば3次元モデリング，隠面消去，レンダリング，スムースシェーディングといった3次元グラフィックに要求される高い計算負荷を分散し並列処理させるシステムを構成できます．

このシステムでは，現実的な3次元グラフィック・サブシステムを，少量のチップで実現することが可能です．またそれらの上で走るソフトウェアは，ANSI標準の3次元グラフィックス・インターフェースPHIGSや，事実上の標準となりつつあるRenderManインターフェース[5]に基づいて当面作成されることになるでしょう．

◈ 参考文献 ◈

(1) IMS G300 color video controller preliminary Data，1988年11月，INMOS Ltd.
(2) 開発が盛んな3次元図形描画コントローラチップ，日経データプロ．
(3) PC Graphics，Electronics，1989年4月号，pp. 84～88.
(4) CADと3次元グラフィックスを融合，NIKKEI ELECTRONICS，1989年3月6日，pp. 108～138.
(5) The RenderMan Interface，BYTE，pp. 267～276.
(6) Processors for 3-D graphics，EDN，1989年3月30日，pp. 97～104.
(7) 第2世代グラフィックプロセッサの活用法，日本AMD，pp. 1～5.
(8) 高速・高機能化で応用進むグラフィックコントローラ/コントロールボードの最新動向，Computer Design，1987年8月号，pp. 25～41.

技術のベースをかためるための CORE BOOKS

アナログ回路のグレードアップ技法

―いかにしてローコストで高性能化を図るか―

中野正次 著　A 5 判，308頁，定価2,400円(税込)

アナログ回路は，OPアンプICの出現により初心者でもある程度の性能の回路を設計することが可能となりました．しかし，精度を要するものとか，同じ機能でも低消費電力化を図りたい，あるいは高速動作に耐えられる回路を実現したいなどと要求スペックが厳しくなるとそう簡単にはいきません．つまり，それ相応の技術と経験が必要とされます．

本書は，永年電子技術に携ってきた著者が，アナログ回路をあらゆる角度から再検討し，いかにしてローコストで高性能を得るかについて，実験的裏付けをもとにわかりやすく解説をしています．

内容

アナログ回路の設計・製作

現実的な回路の作り方と実際の設計法

A 5 判248頁

定価1,700円 (税込)

青木英彦著

CONTENS

CQ出版社　〒170 東京都豊島区巣鴨1-14-2　☎03-947-6311　振替 東京0-10665

'89 最新 半導体規格表

各巻共
A5判
定価 1,000円
(税込)

改訂● 最新 トランジスタ規格表

市場で入手可能な製品のうち約5000本のトランジスタを網羅．デジトラなどの面実装品も掲載．

● 最新 トランジスタ互換表

国内メーカにおけるEIAJ登録トランジスタについて，日本の主要８社におけるそれぞれの互換製品を表形式で掲載．

● 最新 ダイオード規格表

国内メーカの多品種あるダイオードを23種類に分類し，詳細な規格と外形情報を見やすく掲載．

● 最新 光半導体素子規格表

国内外のLEDランプ，LEDディスプレイ，フォト・ダイオード，フォト・トランジスタ，フォト・インタラプタ，フォトICなど一般的な光半導体素子を集大成．

● 最新 メモリIC規格表 1（RAM編）

国内外メーカのスタティックRAM，ダイナミックRAMの最新データを掲載．

● 最新 メモリIC規格表 2（ROM編）

国内外メーカのP-ROM，UV-EPROM，EEPROM，漢字ROMの最新データを掲載．

● 最新 マイコン周辺LSI規格表

8/16ビットCPUの周辺LSIの中で利用の多い120品種を２頁単位で簡潔に説明．セカンド・ソース一覧も掲載．

● 最新 電源用IC規格表

国内外のシリーズ・レギュレータ，スイッチング・レギュレータ，基準電圧源，電圧監視・保護回路，その他電源関連ICを多数掲載．

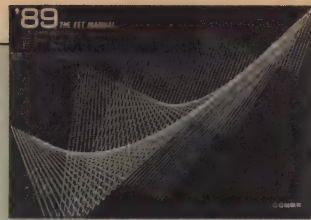
規格表常備店一覧

（＊はデータブックの常備店も兼ねています）

店名	住所
●北 海 道	
紀伊國屋札幌店	札幌市中央区南1条西1丁目42-2
旭屋書店	札幌市中央区南3条西4丁目エイトビル
森文化堂	函館市松風町3-13
旭屋書店	苫小牧市表町6-2-1サンプラザ6F
三省堂	旭川市１条通８丁目右１号AMS7F
●青 森 県	
紀伊國屋書店弘前店	弘前市土手町126-1
●岩 手 県	
東山堂書店	盛岡市中の橋通1-5-23
東山堂ブックセンター	盛岡市大通り1-2-1(サンビル内)
●宮 城 県	
＊金港堂	仙台市一番町2-3-26
金港堂ブックセンター	仙台市一番町3-11-15ジャスコ6F
高山書店	仙台市一番町3-8-15
アイエ駅前店	仙台市中央1-10-1宮城ビル
丸善仙台支店	仙台市国分町1-5-26
●秋 田 県	
三浦書店	秋田市中通2-1-5
加賀谷書店	秋田市中通2-6-51
●福 島 県	
西沢書店	福島市大町7-20
コルニエいわせ	福島市栄町10-11コルニエツタヤビル
ヤマニ書房	いわき市平2-7
●茨 城 県	
川又書店	水戸市泉町2-2-31
田所書店	日立市鹿島町1-12-9
筑波大会館書籍部	新治郡桜村妻木天久保筑波大会館内
●栃 木 県	
新星堂宇都宮店	宇都宮市曲師町6-2オリオン通り
落合書店オリオン支店	宇都宮市江野町3-9
●群 馬 県	
煥乎堂	前橋市本町1-3-4
新星堂高崎店	高崎市連雀町５
戸田書店藤岡店	藤岡市中栗須286
●埼 玉 県	
須原屋	浦和市仲町2-3-20
押田謙文堂	大宮市宮町1-18
須原屋蕨店	蕨市塚越1-4-2
黒田書店	川越市脇田町3-6
●千 葉 県	
キディランド	千葉市新千葉1-1駅ビル3F
旭屋船橋店	船橋市本町7-1-1東武5F
芳林堂津田沼店	船橋市前原西2-18-1パルコ5F
新星堂柏店	柏市柏1-2-31
●東 京 都	
三省堂本店	千代田区神田神保町1-1
＊書泉グランデ	千代田区神田神保町1-3
明正堂	千代田区外神田1-17-15秋葉原デパート
＊万世書房	千代田区外神田1-14-2
＊電波堂	千代田区外神田1-10-11 東京ラジオデパート2F
ADOショップ	千代田区外神田4-4-1
＊丸善	中央区日本橋2-3-10
東京旭屋書店	中央区銀座5-2-1東芝ビル
＊八重洲ブックセンター	中央区八重洲2-5-1
旭屋書店渋谷店	渋谷区宇田川町23-3第一勧銀共同ビル
三省堂書店渋谷店	渋谷区渋谷2-21-12東急文化会館
紀伊國屋書店渋谷店	渋谷区道玄坂1-2-2東急ビル
＊大盛堂書店	渋谷区神南1-22-4 日本綜合ブックセンター
東大本郷	文京区本郷7-3-1
明屋書店	品川区西五反田1-3-8五反田CSビル
雄峰堂五反田店	品川区西五反田7-22-17TOC内B1
ヤマトサンカマタ	大田区西蒲田7-68-1西館６階
栄松堂蒲田店	大田区西蒲田7-69-1
東工大	目黒区大岡山2-12-1
紀伊國屋書店高島屋店	世田谷区玉川3-17-1玉川高島屋内
紀伊國屋書店	新宿区新宿3-17-7
三省堂書店新宿西口店	新宿区西新宿1-1-3小田急スカイタウン
西武ブックセンター	新宿区歌舞伎町1-30-1ぺぺ６F
東京理科大生協	新宿区神楽坂1-3
芳林堂高田馬場店	新宿区高田馬場1-26-5
明屋書店	中野区中野5-52-15ブロードウェイ内
東京旭屋書店池袋店	豊島区西池袋1-1-25東武10階
芳林堂書店	豊島区西池袋1-17-7
三省堂池袋店	豊島区南池袋1-28-1パルコ内
リブロ池袋店	豊島区南池袋1-28-1
弘栄堂書店	武蔵野市吉祥寺南町1-1-24
パルコ吉祥寺店	武蔵野市吉祥寺本町1-5-1
第九書房	三鷹市下連雀3-35-22
オリオン書房ウイル店	立川市曙町2-1-1
啓文堂書店府中店	府中市9065-1
電通大生協	調布市調布ヶ丘1-5-1
鉄生堂書店	八王子市旭町2-12
くまざわ本店	八王子市旭町2-13
有隣堂町田店	町田市原町田6-6-14
久美堂小田急町田店	町田市原町田6-12-20
●神奈川県	
有隣堂	横浜市中区伊勢佐木町1-4-1
有隣堂横浜トーヨー街店	横浜市西区北幸1-8
有隣堂東口店	横浜市西区高島2-16
ブックポート203	横浜市鶴見区中央3-6-25
平坂書房ウォーク店	横須賀市若松町1-5ウォーク横須賀内
文学堂	川崎市川崎区東田町5-11
ブックセンター文教堂	川崎市高津区溝ノ口305東急ストア
有隣堂藤沢店	藤沢市南藤沢2-1-1ダイヤモンドビル内
文教堂六会店	藤沢市亀井町4-15-6
サクラ書店駅ビル店	平塚市宝町1-1平塚ステーションビル4F
東海大湘南	平塚市北金目1117
文教堂四之宮店	平塚市四之宮1131-1
八小堂書店	小田原市栄町1-4-5
伊勢治書店	小田原市栄町2-13-3
文教堂中央林間店	大和市中央林間4-4330中央林間駅ビル
有隣堂厚木店	厚木市中町2-6三成ほていやビル
文教堂星ヶ丘店	相模原市千代田6-3千代田プラザ
伊勢原秦野店	秦野市曽屋5909
●新 潟 県	
北光社	新潟市古町通６
＊紀伊國屋書店	新潟市万代町1-3-30シルバーホテル内
長岡書房	長岡市東坂之上町2-4-9
●富 山 県	
瀬川書店	富山市総曲輪3-7-1
文苑堂	高岡市末広町40
●石 川 県	
うつのみや片町店	金沢市片町2-1-7
王様の本	石川郡野々市町扇ヶ丘4-3
●福 井 県	
品川書店	福井市順化1-1-9
●山 梨 県	
貴川朗月堂	甲府市貢川本町1429
●長 野 県	
平安堂長野店	長野市南千歳町841中谷ビル内
ブックスロクサン	松本市深志1-22-11昭和ビル内
鶴林堂	松本市大手3-3-2
平安堂	飯田市銀座4-10
●岐 阜 県	
自由書房	岐阜市神田町4-9
大衆書房	岐阜市神田町6-6

CQ出版社

好評発売中

〔CQ出版社の半導体規格表は下記書店にてお求めください〕

●最新FET規格表

国内メーカすべてのFET詳細規格と，海外メーカ大手8社におけるパワーMOS FETの規格も掲載.

●最新C-MOS IC規格表

国内外メーカにおける4000/4500B，74H，40Hシリーズのc-MOSタイプのロジックICをすべて掲載．各社セカンド・ソースも併記.

●最新電力用素子規格表

国内メーカの各種サイリスタ，双方向サイリスタ，バリスタ・ダイオード，UJT，各種モジュールなどを掲載.

●最新インターフェース素子規格表

国内外メーカの各種ドライバ/レシーバ用IC，フォト・カプラ，電圧・電流インターフェース素子などを掲載.

●最新TTL IC規格表

国内外メーカのTTL IC(ノーマル，LS，ALS，F，S，AS，ALS/AS1000番台）規格とピン・コンパチブルなHC-MOS(HC，HCT)および各社のセカンド・ソースを併記．さらにPAL（プログラマブル・アレイ・ロジック)の主要規格も掲載.

●最新モノリシックOPアンプ規格表

国内外メーカ約30社のモノリシックOPアンプを一挙に掲載．同時にセカンド・ソースも紹介.

●最新ハイブリッドOPアンプ規格表

国内外メーカ十数社のモジュールを含むハイブリッドICを掲載.

●最新産業用リニアIC規格表「PART-1」

OPアンプを除く国内外の代表的メーカのコンパレータ，国内製品を中心としたアナログ・スイッチ，*V-F/F-V*コンバータ，タイマなどの規格とセカンド・ソース製品を掲載.

●最新産業用リニアIC規格表「PART-2」

OPアンプを除く国内メーカを中心とした低周波電圧増幅器，低周波電力増幅器，広帯域/ビデオ増幅器，計測用アンプ，差動増幅器など各種増幅器用ICの特性と応用回路例を掲載.

●最新A-D/D-Aコンバータ規格表

国内外メーカのA-D/D-AコンバータICを並列比較型A-D，逐次比較型A-D，積分型A-D，Bi電流出力D-A，基準電源内蔵型D-A，出力アンプ内蔵型D-Aなどに分類し，多数掲載.

書店	所在地
●静岡県	
吉見書店	静岡市七間町3
静岡谷島屋	静岡市呉服町2-5-5
江崎書店	静岡市呉服町2-6-8
マルサン宝塚店	沼津市上本通り8
文華堂	富士市吉原2-3-17
岳陽堂	富士市本町12-8
戸田書店	清水市銀座4-6
戸田書店佐鳴台店	浜松市佐鳴台1-11-10
戸田書店幸町店	浜松市幸4-4-1
●愛知県	
日進堂北店	名古屋市瑞穂区桜見町1-3
ちくさ正文館	名古屋市千種区覚王山通り2-2
名古屋大生協	名古屋市千種区不老町1
正文館	名古屋市東区東片端3-1
丸善ブックメイツ	名古屋市中区錦3-15-13
＊名古屋丸善店	名古屋市中区栄3-2-7
三省堂名古屋店	名古屋市中村区名駅1-1-2
近鉄星野書店	名古屋市中村区名駅1-2-2近鉄ビル内
杁中三洋堂刈谷店	刈谷市桜町1-24
精文館トヨダ	豊田市山の手8-92
シビコ正文館	岡崎市康正通西2-20-2
精文館	豊橋市広小路1-6
豊川堂	豊橋市呉服町40
栄進堂小牧店	小牧市岩崎町藤塚372-8
●三重県	
別所書店第11ビル店	津市栄町3-14第11ビル内
シェトワ白揚	四日市市芝田1-6
●京都府	
＊京都駸々堂京宝店	京都市中京区河原町三条下ル大黒町
オーム社書店京都支店	京都市中京区河原町四条上ル
淳久堂京都	京都市下京区四条通柳馬場東入立売東町20-1
＊アバンティブックセンター	京都市南区東九条西王町31アバンティ6F
●大阪府	
旭屋難波店	大阪市南区難波新地6-12ナンバCITY内
駸々堂心斎橋店	大阪市南区末吉橋通3-14
旭屋本店	大阪市北区曽根崎2-12-6
＊紀伊國屋書店梅田店	大阪市北区芝田1-1-3
オーム社書店	大阪市北区堂島中1毎日大阪会館
大阪工業大学生協	大阪市旭区中宮5-12-1
旭屋アベノ店	大阪市阿倍野区阿倍野筋1-6-6
かつらぎ	大阪市福島区大開1-5-26
ニノミヤパーツ	大阪市浪速区日本橋筋4-54
関西大生協	吹田市千里山東3-10-1
旭屋書店京阪守口店	守口市河原町3京阪守口百貨店内
●兵庫県	
漢口堂星電店	神戸市中央区三宮町1-4-10阪急共栄物産ヒスアンドハース店内
流泉書房	神戸市中央区三宮町1-5-27
＊淳久堂書店	神戸市中央区三宮町1-6-18三宮センター街ニューセンター地階
日東館書林	神戸市中央区三宮町3-3-1
丸善神戸支店	神戸市中央区元町通1-4-12
海文堂	神戸市中央区元町通3-5-10
ジュンク堂サンパル店	神戸市中央区雲井通5サンパル3F
小山助学館明石店	明石市大明石町1-1-23明石ステーションデパート2F
新興書房	姫路市直養町89-5
誠心堂	姫路市光源寺前町10
●和歌山県	
宮井平安堂	和歌山市本町1-7
●鳥取県	
富士書店	鳥取市末広温泉町164
今井書店本通り店	米子市四日市町86
●島根県	
今井書店	松江市殿町63
●岡山県	
細謹舎書店	岡山市表町2-1-37
紀伊國屋岡山店	岡山市中山下2-2-1
●広島県	
広文館本通店	広島市中区本通3-5
丸善	広島市中区本通5-8
金正堂	広島市中区本通5-9
＊紀伊國屋広島店	広島市中区基町6-27広島センタービル6F
フタバCP卸部	広島市西区観音本町2-8-22
サントーク広文館	福山市三之丸町サントーク内
●山口県	
文栄堂書店	山口市道場門前1-3-11
●徳島県	
小山助学館	徳島市一番町3-22
小山助学館徳島駅東口店	徳島市寺島本町東3-32-8
●高知県	
金高堂	高知市帯屋町1-13-14
●福岡県	
紀伊國屋書店	福岡市中央区天神1-11-11天神コアビル
りーぶる天神	福岡市中央区天神4-4-11
金栄堂	北九州市小倉北区魚町2-4-6
旭屋北九州店	北九州市八幡西区黒崎1-1-1そごう7F
菊竹金文堂	久留米市中央町34-7
たがみ書店	久留米市東町25-9
●佐賀県	
金華堂	佐賀市白山町2-5-19
●長崎県	
好文堂	長崎市浜町8-29
●熊本県	
まるぶん書店	熊本市上通5-1
紀伊國屋書店	熊本市下通1-7-18
●大分県	
本町晃星堂	大分市中央町1-1-17
●宮崎県	
田中書店	宮崎市橘通東3-1-6
●鹿児島県	
坂口金海堂	鹿児島市東千石町17-1
●沖縄県	
球陽堂書房	那覇市牧志3-2-5

〒170 東京都豊島区巣鴨1-14-2 ☎03-947-6311 振替 東京0-10665

エレクトロニクス界をリードする専門図書

情報処理試験合格のための
プログラム技法の徹底研究
情報処理技術者教育研究会編/A5判 128頁/*900円 〒210円

ソフト 一般

システム操作入門から信号処理への応用まで
オブジェクト指向と Smalltalk
小林史典/A5判 192頁/定価1,700円(税込) 〒260円

マイコン時代の波形処理技術の基礎と応用
ディジタル信号処理入門
三上直樹/A5判 184頁/*1,500円 〒260円

データ構造&アルゴリズムの設計と構造化プログラミング
実践ソフトウェア作法
インターフェース編集部編/B5判 232頁/*1,500円 〒260円

パーソナル・コンピュータによる系の解析テクニック
連続系シミュレーション
小池慎一/A5判 272頁/定価2,060円(税込) 〒260円

システムの内部構造とそのプログラミング
解析 マルチタスク
佐々木 茂/A5判 176頁/*1,500円 〒260円

基礎概念への最新おもしろガイド
ソフトウェア考現学
萩谷昌己/A5判 184頁/*1,300円 〒260円

モジュールからコルーチンまでの詳細
Modula-2 文法入門
中村和郎/A5判 168頁/*1,300円 〒260円

8086による浮動小数点演算の実際と8087の使い方
数値演算入門
大貫広幸/A5判 400頁/*2,600円 〒310円

浮動小数点演算入門から高速演算プログラミングまで
数値演算プロセッサ
インターフェース編集部編/B5判 272頁/*1,800円 〒260円

CP/M, MS-DOS, OS-9, UNIX, OS 一般

MS-DOSソフトの移植からマルチタスク・プログラム作成まで
OS/2 プログラミング
インターフェース編集部編/B5判 176頁/*1,600円 〒260円

16ビット・マイコン用DOSの機能，仕組み，使い方
CP/M-86 入門
北原拓也/A5判 184頁/*1,400円 〒260円

Z80プログラミングの効率を上げるサブルーチン・ライブラリ
CP/M 活用モジュール・ハンドブック
伊藤英明/B5判 240頁/*1,600円 〒260円

作りながら学ぶ
マイコン設計トレーニング
神崎康宏/B5判 280頁/*1,500円 〒310円

CP/M-86と MS-DOS を比較
MS-DOS 入門
北原拓也/A5判 208頁/*1,500円 〒260円

ソフトマインド(2)
OS-9 & 6809 活用プログラミング
有吉 久/B5判 192頁/*1,600円 〒260円

マイコンピュータ No.25
OS-9/68000 の研究
B5判 168頁/*950円 〒260円

Unix/Ada によるソフトウェア構成管理
プログラムのチーム開発入門
Wayne A. Babich 著 菊池豊彦(訳)/
A5判 192頁/*2,000円 〒260円

シェル/C 言語/開発ツールを使いこなす
UNIX プログラミング実践編
金崎克己/A5判 294頁/*1,900円 〒260円

ワークステーションとのつきあいかた
パーソナル UNIX 道具考
祐安重夫/A5判 208頁/*1,500円 〒260円

Unix 流シェルのプログラムと使い方
MS-DOS 用 Shell の実現
Allen Holub 著 横山和由(訳)/A5判 336頁/
定価2,700円(税込) 〒310円

システム操作入門から信号処理への応用まで
オブジェクト指向と Smalltalk
小林史典/A5判 192頁/定価1,700円(税込) 〒260円

パソコンの利用技術

電子回路とパソコンによる計測技術ノウハウ
物理計測システム実用設計
物理教材研究会編/A5判 280頁/定価2,000円(税込) 〒260円

標準入出力インターフェースの規格・使い方・設計ノウハウ
最新 SCSI マニュアル
インターフェース編集部編/B5判 192頁/*1,600円 〒260円

パソコン信号計測ソフト&ハードの基礎と実際
A-D 変換を使いこなす
竹本 晃・稲村 浩共著/A5判 328頁/*1,800円 〒260円

マイコンピュータ No.24
ROM 化プログラムと開発環境
B5判 176頁/*950円 〒260円

標準ディジタル・バスの使い方から設計法まで
IEEE-488(GPIB)とその応用
岡村廸夫/A5判 272頁/*1,600円 〒260円

計測システムにおけるマイコン/パソコン活用技術
科学計測のための波形データ処理
南 茂夫編著/A5判 240頁/*1,900円 〒260円

エレクトロニクスの基礎知識と実践へのアプローチ
パソコン・インターフェース考
佐藤清忠/B5判 180頁/*1,500円 〒260円

パーソナル・コンピュータによる系の解析テクニック
連続系シミュレーション
小池慎一/A5判 272頁/定価2,060円(税込) 〒260円

CQ出版社 〒170 東京都豊島区巣鴨1-14-2 ☎03-947-6311 振替東京0-10665

(*印のものは消費税が加算されます)

各社データブック&マニュアル

データブック＆マニュアルをお求めの方へ!!
各社データブック，マニュアルをお求めの際は，下記の書名に☑印をつけて，お近くの書店または小社営業部へ直接お申し込みください(小社へご注文の節は定価に送料を添えてお送りください).
なお，品切れになる場合もあります．その際はご容赦ください．

●サンヨー

書名	定価(送料)円 (税込)	頁
□ '88-'89 三洋半導体データブック 〈オーディオ用MOS集積回路編〉	*2500(310)	512
□ '89 三洋半導体データブック 〈オーディオ用混成厚膜集積回路編〉	*2500(310)	440
□ '89 三洋半導体データブック 〈映像機器集積回路編〉	*2500(310)	528
□ '89 三洋半導体データブック 〈ダイオード，サイリスタ編〉	2500(310)	582
□ '89-'90 三洋半導体データブック 〈マイクロコンピュータ Vol.1 8/4ビット LCD マイコン編〉	2500(310)	
□ '89-'90 三洋半導体データブック 〈マイクロコンピュータ Vol.2 8/4ビットマイコン，ゲートアレイ編〉	2800(310)	

●新日本無線

書名	定価(送料)円 (税込)	頁
□ '89 半導体データブック　半導体個別素子	*1200(260)	179
□ '89 半導体データブックCMOS IC	*2800(310)	415
□ '88 半導体データブック　バイポーラIC	*2800(310)	516

●ソニー

ソニー半導体集積回路データブック1989

書名	定価(送料)円 (税込)	頁
※□ メモリ	1339(310)	260
※□ オーディオ	3214(360)	498
※□ テレビ	4017(360)	616
※□ CCD撮像素子周辺IC	3605(360)	577
※□ ビデオ	5088(410)	836
※□ コンパクトディスク	3111(360)	489
※□ シングルチップマイクロコンピュータ	2513(310)	372
※□ 個別半導体	2225(310)	330
※□ マイクロプロセッサ	3317(360)	479
※□ 産業用	2287(310)	431
※□ フロッピーディスク/ハードディスク	2657(310)	339
※□ 通信システム用IC	1318(260)	160

●日立

日立データブック

書名	定価(送料)円 (税込)	頁
※□ DIODE '88	1751(310)	388
※□ ICメモリ データブック '88	4635(410)	884
※□ 情報産業用リニアIC '88	3399(360)	676
※□ オプトデバイス '88	2575(310)	360
※ 8ビットシングルチップ マイクロコンピュータ '88	4120(360)	1112
※□ マイクロコンピュータサポートシステム '89	4120(310)	328
※□ 6800, 68000 シリーズ マイクロプロセッサ/周辺LSI '89	2987(310)	750
※□ 通信用半導体 '89	6180(360)	764
※□ パワーMOS FET '89	4738(360)	628

●富士通

富士通半導体デバイスDATA BOOK 1989

書名	定価(送料)円 (税込)	頁
※□ シリコン　トランジスタ	*1800(360)	694
※□ 光半導体	*1000(260)	194
※□ 標準ロジック	*3400(460)	1372
※□ メモリ	*4000(510)	1906
※□ ASSP/汎用リニアIC	*3800(460)	1672
※□ マイクロプロセッサ	*3400(460)	1334
※□ マイクロコントローラ	*4000(510)	2076
※□ GaAs FET	*1400(310)	406
※□ パッケージ	*1200(310)	399
※□ 通信用IC	*2600(410)	912

●モトローラ

書名	定価(送料)円 (税込)	頁
※□ ハイスピードCMOSロジックデータ	*3000(360)	761
※□ CMOS LOGIC DATA (英文)	*2500(310)	526
※□ CMOS-NMOS SPECIAL FUNCTIONS DATA	*1100(260)	373
□ M68000マイクロプロセッサ・ユーザーズ・マニュアル	*2500(260)	210
□ MC68020 ユーザーズ・マニュアル	*3500(310)	349
□ FAST AND LS TTL DATA (英文)	*2500(310)	622
※□ FAST & LS TTL データ・ブック (和文)	*2000(310)	554
※□ LINEAR AND INTERFACE INTEGRATED CIRCUITS (英文)	*3800(360)	1534
※□ MC68030 USER'S MANUAL(英文)	*4300(360)	
※□ M68000 USER'S MANUAL(英文)	*2000(310)	
※□ MECL DEVICE DATA(英文)	*2400(310)	
※□ FACT DATA(英文)	*1500(260)	
□ MC68030 32ビット・シングル・ボード・マイクロコンピュータ ユーザーズマニュアル	*1900(310)	
□ M68000 FAMILY REFERENCE	2060(310)	
□ MVME 133 VMEmodo 32ビット　モノボード マイクロコンピュータ ユーザーズマニュアル	*1200(260)	128

●VME MEMBER

書名	定価(送料)円 (税込)	頁
□ VMEbusアーキテクチャ・マニュアル	*3600(310)	263
□ VMEプロダクツ イン ジャパン 1989 Vol.1	*5000(310)	222

●テキサスインスツルメンツ

書名	定価(送料)円 (税込)	頁
※□ ALS/AS Logic Circuits DataBook 1987	*3000(410)	954

ご住所	〒	お名前	

書名	定価(送料)円(税込)	頁
※□ AS8XX/ACT88XXファミリー ビット スライス アプリケーションノート 1988	*1500(260)	114
※□ 高速CMOSロジック データブック '88	*3000(360)	723
※□ TMS34010ユーザーズ・マニュアル 第1部 (アーキテクチャ/ハードウェア編)	*2500(310)	300
※□ TMS34010ユーザーズ・マニュアル 第2部 (命令セット編)	*2500(310)	284
※□ TMS32020ユーザーズ・マニュアル	*1500(300)	256
※□ TMS320C25ディジタル・シグナル・プロセッサ ユーザーズ マニュアル	*2500(310)	310
※□ インターフェイス・サーキット アプリケーションノート	*1000(260)	188
※□ TMS9914A -GPIBアダプタ ユーザーズ マニュアル	*800(210)	92
※□ インターフェイス サーキット データブック '89	*3000(360)	866
※□ ディスプレイ ドライバ IC データブック '89	*1500(310)	326
※□ リニア サーキット データブック1989	*3000(410)	1140
□ TTL STD, LS, S(汎用ロジック) データブック	3090(360)	926

●アナログ・デバイセズ

書名	定価(送料)円(税込)	頁
□ アナログ・デバイセズ データブック	*4000(510)	2384
□ ADSP-2100 ユーザーズ・マニュアル	*2100(310)	200

●日本AMD

書名	定価(送料)円(税込)	頁
□ 32ビット RISC マイクロプロセッサ ユーザーズ マニュアル(日本語版) 上巻	*2000(260)	162
□ 32ビット RISC マイクロプロセッサ ユーザーズ マニュアル(日本語版) 下巻	*2500(260)	292

●日本バー・ブラウン

書名	定価(送料)円(税込)	頁
□ プロダクト データブック 1989	*3600(460)	1482

●ナショナル セミコンダクタージャパン

書名	定価(送料)円(税込)	頁
□ リニアICデータブック	*4500(460)	1259

●インテル

書名	定価(送料)円(税込)	頁
※□ MCS-51ファミリ・ユーザーズ・マニュアル (第2版)	*6000(260)	190
※□ MCS-51マクロアセンブリ言語ユーザーズ・ガイド	*3800(310)	282
□ MCS-96ユーザーズ・マニュアル	*3600(260)	184
□ MCS-96マクロ・アセンブラ・ユーザーズ・ガイド	*4500(260)	184
※□ 386TM マイクロプロセサ・ハードウェア・リファレンス・マニュアル	*5000(260)	152
※□ 80386プログラマーズ・リファレンス マニュアル	*6000(310)	340
※□ 80386システム・ソフトウェア ライターズ・ガイド	*4800(260)	114
※□ 80387プログラマーズ・リファレンス・マニュアル	*5000(260)	164
※□ MCS-85マイクロコンピュータ・ユーザーズ・マニュアル	*4500(310)	301
※□ 8080/8085 アセンブリ言語プログラミング・マニュアル	*3000(260)	370
※□ 80286 プログラマーズ・リファレンス・マニュアル	*6000(310)	412
※□ 80286 オペレーティング・システム・ライターズ・ガイド	*4500(260)	166
※□ 80286 ハードウェア・リファレンス・マニュアル	*5000(260)	162
□ ASM286アセンブリ言語リファレンス・マニュアル	*8600(310)	276
□ ASM286マクロ・アセンブラ操作説明書	*3000(210)	72
□ 80286ユーティリティ・ユーザーズ・ガイド	*5000(260)	176
※□ PL/M-286ユーザーズ・ガイド	*7600(260)	224
□ 80186/188 USER'S MANUAL=HARDWARE REFERENCE(英文)	*4200(260)	350
□ 80186/188 USER'S MANUAL=PROGRAMMER'S REFERENCE(英文)	*4200(260)	376
※□ PL/M-86 ユーザーズ・ガイド	*4800(260)	144
※□ インテルMULTIBUS I 仕様説明書	*3000(260)	106
※□ インテルMULTIBUS 2バス・アーキテクチャ仕様説明書	*5000(260)	148
※□ iRMX86オペレーティング・システム入門	*1600(210)	120
※□ ASM86 アセンブリ言語リファレンス・マニュアル	*6000(310)	286
※□ iAPX86ファミリ・ユーザーズマニュアル	*7600(360)	715
※□ Microprocessor and Peripheral Handbook Volume I -Microprocessor 1989(英文)	*2600(360)	1414
※□ Microprocessor and Peripheral Handbook Volume II -Peripheral 1989(英文)	*2600(310)	1210
※□ コンポーネント品質/信頼性ハンドブック	*7000(260)	200

*印の価格は消費税が加算されます.

●インテル・ジャパン㈱発行図書で, 上記掲載図書以外の注文は受注注文〔インテル・ジャパン㈱からの取り寄せ〕品となり, 多少時間がかかることがございますので, あらかじめご了承ください.

●各社とも改訂中のものは掲載されていません.

データブック 常備店一覧

書店名	所在地	電話
金港堂本店	仙台市	☎022-225-6521
紀伊國屋書店 新潟店	新潟市	☎0252-41-5281
書泉グランデ	千代田区	☎03-295-0011
八重洲ブックセンター	中央区	☎03-281-1811
日本橋丸善書店	中央区	☎03-272-7211
大盛堂書店	渋谷区	☎03-463-0511
東大本郷生協	文京区	☎03-811-5407
万世書房	千代田区	☎03-255-0605
電波堂	千代田区	☎03-255-8539
有隣堂横浜トーヨー街店	西区	☎045-311-6265
名古屋丸善書店	中区	☎052-261-2251
紀伊國屋書店	北区	☎06-372-5821
駸々堂京宝店	中京区	☎075-223-1003
アバンティブックセンター	南区	☎075-671-8987
淳久堂三宮店	中央区	☎078-392-1001
紀伊國屋書店 広島店	中区	☎082-225-3232

(上記の書店では小社の取り扱っているデータブックのうち※印のものを展示・販売しております)

ご住所	〒	お名前	

FINGERTIP AD INDEX

インターフェースでは、読者各位に掲載広告を有効活用していただくために、50音別広告索引とともに製品別広告索引を設置しています。これは、小誌だけでなく姉妹誌トランジスタ技術と同一の区分法に基いています。さらに、小社発行半導体情報誌DATUMとも歩調をそろえています。
具体的には、大分類として18項目さらに中分類と分化して掲載しています。
しかし、読者各位から、4色広告あるいは2色広告を別として、白黒広告にさらに簡便な見出しがあればとの声が寄せられています。そこで、そのご要望にお応えしてFINGERTIP AD INDEXを設けました。これは、掲載広告に大分類に基いた見出しをつけて検索を容易にするものです。
用途に応じて、製品別広告索引とFINGERTIP AD INDEXをお使い分けいただければ幸甚です。

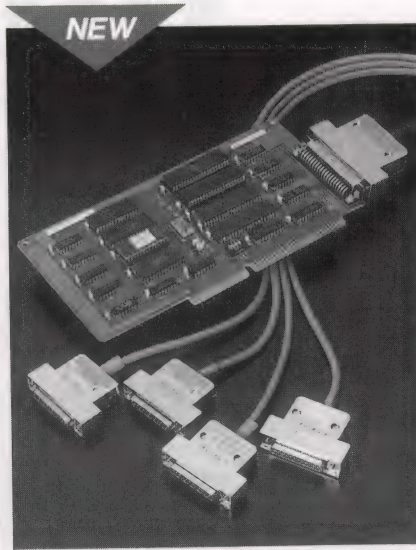
分類	内容	
半導体	ディスクリート、CPU/MPU、メモリIC/LSI、周辺IC/LSI、その他のディジタルIC/LSI、増幅器IC/LSI、その他のアナログIC/LSI、光関連IC/LSI、カスタム・セミカスタムIC/LSI、その他の半導体。	A
ボード/モジュール	CPUボード/モジュール、メモリボード/モジュール、周辺ボード/モジュール、その他のディジタル処理用ボード/モジュール、アナログ処理用ボード/モジュール、光関連ボード/モジュール、各種標準規格ボード、シングルボードコンピュータ、など。	B
表示部品	電子管、ディスプレイモジュール、LEDモジュール、LCDモジュール、など。	C
受動部品	抵抗器、ダブルバランスドミキサ、コンデンサ、トランス、コイル、フィルタ、スイッチ、リレー、モータ、ブロア、ファン、ランプ、電球、センサ、振動子、共振子、発振子、EMI対策用部品、その他の受動部品。	D
実装部品	プリント回路基板、積層板、コネクタ、放熱器、ケーブル、線材、筐体、シャシン、ソケット、端子部品、パッケージ、各種実装部品、その他の実装部品。	E
電子材料	セラミック、ガラス、フォトレジスト、コーティング材料、接着材料、磁気材料、絶縁材料、導電材料、その他の電子材料。	F
電源機器	電源装置、組込用電源機器、各種電池、その他の電源機器。	G
計測機器	電流/電圧/電力計測器、回路素子/電磁性体計測器、周波数/時間計測器、伝送特性計測器、発振/信号発生器、電波計測器、AV計測器、光計測器、データ通信計測器、環境計測器、データロガー、記録機器、IC/LSIおよびボード・テスタ、その他の計測機器。	H
半導体、電子部品製造関連装置	半導体製造装置、その他の電子部品製造関連装置。	I
その他の電子機器	音響/映像機器、音声合成機器、その他の電子機器。	J
コンピュータ	パーソナルコンピュータ、ミニコンピュータ、ワーク・ステーション、その他のマイクロコンピュータ装置、CADシステム、画像処理システム、LANシステム、VANシステム、FA/LAシステム、その他のコンピュータ。	K
開発支援装置	インサーキットエミュレータ(ICE)、デバッガ、マイコン開発支援装置、ロジック・アナライザ、PROM/PAL書込器など。	L
周辺機器	印刷関連機器、画像関連機器、記憶関連機器、通信/伝送関連機器、モデム、切替器、周辺機器関連製品および消耗品、その他の入力関連機器、その他の出力関連機器、その他の周辺機器。	M
ソフトウェア	OS、言語、AIソフト、ワードプロセサソフト、簡易言語、業務ソフト、グラフィックソフト、CADソフト、技術計算ソフト、ユーティリティソフト、統合ソフト、通信用ソフト、データベースソフト、開発支援ソフト、医療ソフト、その他のソフトウェア。	N
その他の材料、部品、装置	その他の材料、部品、装置、治具、工具など。	O
流通	各種電子機器販売、各種電子部品販売、各種ソフトウェア販売、レンタル、リース、システムハウスなど。	P
情報・教育	情報サービス、教育、講習会、出版、書籍など。	Q
案内	業務案内、催物告知、人材募集など。	R

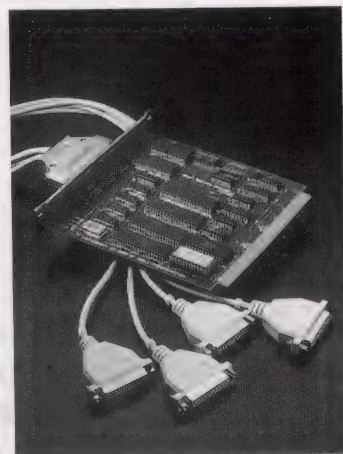
B

CHANNEL-4 Multi Server

FMR用、IBM PC/AT用新登場!!

NEC PC-9801シリーズで好評を得た、多重回線用通信制御ボード、マルチサーバ・チャンネル-4が富士通FMR50/60/70シリーズ(Panacom Mシリーズ)、IBM PC/AT、AX対応として新登場します。多重回線用通信制御ボード、マルチサーバ チャンネル-4が①CPUクロック周波数を6MHzとし②FAの厳環境下での使用も踏まえケーブル・シールドは、D-Subコネクタ及び取付金具によりパソコン本体ケースにアースされる等、内容を充実し、パソコン市場を代表するNEC、富士通、IBMマシンへの対応を実現しました。また マルチ回線BBSホストシステム用としての整合性等何れをとってもユーザーオリエンテッドなインテリジェントタイプのRS232C拡張用ボードです。

CHANNEL-4/F
定価¥98,500(消費税除く)

CHANNEL-4/I
定価¥110,000(消費税除く)

CHANNEL-4/N
定価¥98,500(消費税除く)

新発売
別売 マルチ・サーバ用OS/2ディバイス・ドライバー
定価¥21,300(消費税除く) 開発元/デジタルシステム

特徴

① ボード上に8ビットCPU Z-80を搭載し、パソコンを通信用線間で、通信データのコントロールを行わせることにより本体ハードウエア受信割込を省略化しています。パソコンCPUの負荷を軽減し、ソフトウエアの簡略化を実現します。

② パソコンとの通信は外部バスを用いた、8ビットパラレル入出力で行われます。通信速度は約20Kバイト/秒、シリアル換算でチャンネル当り40Kbpsの応答速度がありリアルタイム、高速通信が可能となります。

③ PS232C制御ラインのオン・オフ通信パラーメータの設定は、全てソフトウエアによるコントロールです。ソフトウエアコマンドは、各機種共通仕様になっています。他機種への変更が容易に行えます。

④ PC-9801シリーズで256枚、FMRシリーズ、PC/ATは16枚まで同時に使用出来ます。

*マルチサーバは、㈱ベルコーポレーションの開発製品です。

■発売元

株式会社

〒110 東京都台東区台東1丁目1番14号千代田機工ビル6F
TEL(03)835-9031(代) FAX(03)835-9034

資料請求No.308

B

NEC PC-9800シリーズ用拡張ボード

認定番号
S89-0107-0

※(特許出願中)

電話多機能化ボード

でんたきボード

EBX-98-601　¥45,000(ソフト付)

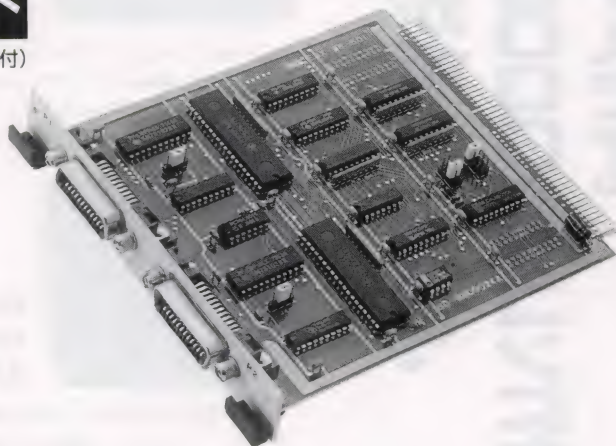
特徴

- 通常人手で行っている電話の発信・着信・転送・ポケットベルの呼び出し・音声メッセージの録音再生などを、パソコンとこのボードが無人で代行します。
- 機能として、発呼、着呼、ダイヤルトーンの判別・発生、フッキング、転送(1回線)※、TEL/FAXの自動判別、外部機器のコントロール(赤外線リモコン)があります。
- 音声データの録音・再生はパソコンのメモリを使います。(1.4Kバイト/秒)
- プッシュホンからのデータ入力、音声自動応答システムに最適です。
- ハンドリングソフト(TDD)付属。

バイポートGP-IBボード

EBX-98-501　¥88,000(ハンドラソフト付)

特徴

- GP-IBコントローラを2ヶ積載。2つのポートが独立に動作します。
- バスアナライズ機能(ソフトウェア対応)。ポート0でデータ転送しながらポート1でバスの信号を観測できます。
- MS-DOS 対応のハンドラソフト BCOM Ver6.2添付。MS-DOSのもとで動くアセンブラ、BASIC、C.FORTRANなど各種言語でプログラムの作成ができます。
- 120Kバイト/秒の高速転送(クロック10Mhzにて)
- 理解を助ける参考書　岡村迪夫著："IEEE-488(GP-IB)とその応用"CQ出版、1988があります。
- RA、RXシリーズ対応。

[でんたきボードの機能を活かすアプリケーションソフト]

- 多機能留守番電話システム**"イルーサー1"**
  ¥68,000("でんたき"ボード付き)
- パーソナルボイスメールシステム**"声の私書箱"**
  ¥158,000("でんたき"ボード付き)(ハードディスクが必要です)

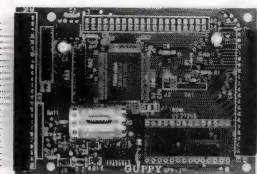
24時間求人案内システム

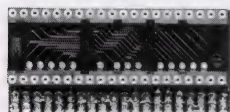
でんたきボード応用の弊社求人案内システム デモ中!
TEL 03-458-0529。応募される方もされない方も是非お試し下さい。

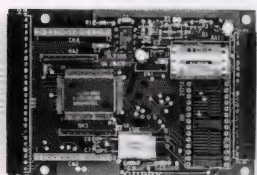
通信と制御の
日本ビジネスシステムズ株式会社
〒140 東京都品川区北品川3-11-13 荏原北品川ビル4F
TEL.03-458-6474　FAX.03-458-6470

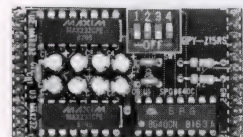
ご用命は下記販売店へ

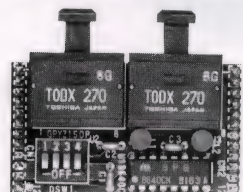
㈱コム	☎ 03-251-1523
佐鳥電機㈱　支店・営業所	☎ 03-452-7491
㈱リョーサン　情報機器専売部	☎ 03-864-5841
システムインNitsukoグループ	☎0425-27-3211
㈱フェーチャーイン　名古屋	☎052-201-2555
サンワサプライ㈱	☎ 03-546-2781

GUPPY® シリーズ●他に類を見ないCPU群と豊富なI/Oカード!

GPY-Z11
Z84C011 CPU CARD
¥24,800

TMPZ84C011AF(Zilog社のZ80A+CTC+CGCの機能内蔵) 入出力ポート8bit 5ch(CPUに内蔵)、全てのポートは10KΩにてプルアップされています。リセットIC(TL7705)使用 6264タイプ(RAM)1個実装し、メモリーバックアップ可能(リチウム電池付) 27256(ROM) GPY-ICE-Z11を装着するとインサーキットエミュレータに接続できます。(GPY-ICE-Z11はオプションです。)

GPY-ICE-Z11
ICEプローブ接続用拡張CARD
¥7,000

ICE(インサーキットエミュレータ)に接続する為の40ピンDIPカードです。

GPY-Z15
Z84C015 CPU CARD
¥24,800

TMPZ84C015AF(Zilog社のZ80A+CTC+SIO+PIOの機能内蔵)/PIOのPA、PB各8bitは10KΩにてプルアップされています/リセットIC(TL7705)使用/6264タイプ(RAM)1個実装し、メモリーバックアップ可能(リチウム電池付)/27256(ROM)/オプションにて、GPY-Z15RS(RS-232C 2ch CARD)、GPY-Z150P2(光リンク2ch CARD)があります。

GPY-Z15RS
RS-232C 2chCARD
¥12,000

RS-232CレベルコンバータにMAX232CPEを使用/ボーレートジェネレータに8640CNを使用(300～9600ボーまで、DIP SWにて設定可能)

GPY-Z150P2
光リンク 2ch CARD
¥26,000

光リンクモジュールにTODX270を2個使用/ボーレートジェネレータに8640CNを使用(300～9600ボーまでDIP SWにて設定可能)

GPY-15
8bit 2ch D/A
コンバータ
CARD
¥14,800

仕様：AD7523JN(DAC)/8255(PPI)/4bitの汎用ボード付

GPY-16
PIO/CTC
CARD
(GPY-01専用)
¥17,800

仕様：Z8420A(PIO)/Z8430A(CTC)/モード2割込可/デイジーチェイン可

GPY-19
RTC CARD
¥14,800

仕様：RTC-62421A(RTC)/タイマ割込可/バッテリーバックアップ付/PST-520C(停電検出用IC)/多目的4bit DIPSW付。

GPY-35
カラー
CRTC CARD
¥39,800

CRTCにRP5C16を使用したカラーCRTCコントローラカードです。最大15色のRGB出力で、80×25文字(640×200ドット)のカラー画面が表示できます。

GPY-36
4Mbit
ROM CARD
¥17,800

1MビットのROMを4個搭載できます。漢字ROMやデータROM等に使用できます。

GPY-13
7セグLED/キー入力コントローラ CARD
¥17,800

最大64キーと16桁LEDの接続可能な8279を使用したインターフェースカードです。

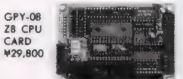
GPY-18
GP-IB CARD
¥24,800

トーカ、リスナ、コントローラの各機能と各種制御機能をもつTMS9914を搭載したGP-IBバスインターフェースカードです。

GPY-22
プリンタインターフェースCARD
¥14,800

セントロニクス準拠のプリンタインターフェースカードです。STB出力をカード内で発生させるので、容易に文字を出力することができます。また、ACKNLG入力により割込(INT)を発生させることができます。

GPY-24
12bit 8ch A/D
コンバータCARD
¥49,800

12bit 8chのA/Dコンバータです。ADCにAD574AJD、S&HにAD582、MPXにAD7503を使用し、入力電圧範囲を0～10Vに選定しています。ユーザオプションにて、0～20V、-5～+5V、-10～+10Vの選択ができます。アナログ用±15Vの電源は、外部供給が必要です。

GPY-08
Z8 CPU
CARD
¥29,800

仕様：Z8671(BASICインタープリタ内蔵CPU)/2764(ROM)、6264(RAM)実装可/6264、1個実装済

GPY-8085
8085CPU
CARD
¥19,800

仕様：8085(CPU)/8155×2(PIO)/2764実装可(ROM)/PST518使用(リセットIC)CLK周波数3MHz

GPY-V20
V20 CPU
CARD
¥29,800

仕様：μPD70108D(V20)/μPD71055C(PIU)/2764実装可(ROM)/6116、6264実装可(RAM)/PST518使用(リセットIC)/μPD71011C(CKG)/CLK周波数5MHz/メモリーBackup端子付

GPY-6303
6303 MPU
CARD
¥24,800

仕様：HD63B03R(MCU)/2732A、2764実装可(ROM)/DC/DCコンバータ内蔵 RS232C 1ch付

GPY-6809
6809 MPU
CARD
¥24,800

仕様：68B09(MPU)/68B21(PIA)
2764実装可(ROM)/6264実装可(RAM)/PST518使用(リセットIC)/リセットSW付/ジャンパ切換により6522A(VIA)使用可

GPY-6321
C-MOS
40bit I/O
CARD
¥19,800

仕様：63B21×2(PIA)/割り込みコントロール用ジャンパ有/68系CPU専用

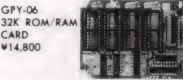
GPY-03
48bit I/O
CARD
¥14,800

仕様：8255(PPI)×2 プルアップ抵抗内蔵

GPY-04
RS-232C 2ch
CARD
¥19,800

仕様：8251A(USART)×2/オンボードレギュレータ内蔵/ボーレート300～19200ボー

GPY-05
VRAM CARD
(GPY-01専用)
¥19,800

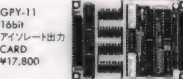
仕様：46505SP-2(CRTC)/CG(2732)内蔵 6116(RAM)/80桁×25行表示可

GPY-06
32K ROM/RAM
CARD
¥14,800

仕様：2764(ROM)又は6264(RAM)を併せて最大4個搭載可/チップごとにバッテリーバックアップ可

GPY-07
64KB
DRAM CARD
¥17,800

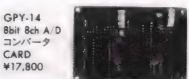
仕様：DRAM/HM50464P-12×2/16KBを1バンクとして4バンク切換可

GPY-10
FDC CARD
¥49,800

仕様：μPD7265(FDC)/FDC9229B(データセパレータ)/3.5インチ、5インチ2HDまで対応

GPY-11
16bit
アイソレート出力
CARD
¥17,800

仕様：TD62003A×3/TLP621×16/最大50V500mA、但し8bitにつき最大1A

GPY-12
17bit
アイソレート入力
CARD
¥17,800

仕様：TLP621×17/ノイズフィルター付/1bitのNMI入力有

GPY-14
8bit 8ch A/D
コンバータ
CARD
¥17,800

仕様：TC5093A(ADC) 変換速度100μS(MAX) EOCによる割込可

GPY-20
DIP SW付
LED CARD
¥14,800

仕様：データ用LED×8/RUN用LED/INT用LED/8ビットDIP SW
※I/Oリード、ライトによりDIP SWのデータを読んだり、LEDを点滅させたりできるのでデバック等に最適。

GPY-25
12bit 1ch
アイソレートD/A
コンバータ
¥49,800

仕様：AD7545LN(DAC)/AD581KH(基準電源)

GPY-23
アイソレート
8bit入力
8bit出力CARD
¥17,800

仕様：入力8bit TLP621×8、ノイズフィルター付 出力8bit TD62003A×2、TLP621×8、最大50V500mA 但し4bitにつき最大1A

その他のカード

GPY-02 ユニバーサルカード………¥2,000
GPY-09K マザーカード3スロットキット………¥8,000
GPY-09F マザーカード5スロット完成品………¥14,800
GPY-17 エクステンションカード………¥14,800
GPY-BOX5 マザーカード付ラック………¥19,800
90(H)×140(W)×177(D)mm
GPY-MON GPY-01、01W、21、41用モニターROM………¥10,000

GPY-180
64180 MPU CARD
¥24,800

GPY-41
Z80A CPU CARD
¥19,800

GPY-01W
Z80A CPU CARD
¥19,800

GPY-01
Z80A CPU CARD
¥14,800

品　名	CPU	LSI	ROM	RAM	リセットIC	メモリバックアップ	価　格
GPY-01	Z80	8255	2732A、2764	6116、6264ジャンパー切換必要	PST-518	可能	¥14,800
GPY-01-C	Z80(CMOS)	8255(CMOS)	同上	同上	同上	同上	¥19,800
GPY-01-R	Z80A	8255	同上(2764付)	同上(6116付)	同上	同上	¥17,800
GPY-01W	同上	8255×2	2764、27128、27256	HM6264ALSP 実装済	TL7705	不可	¥19,800
GPY-21	同上	8255 8253	同上	HM6264付	同上	同上	¥19,800
GPY-41	同上	8420A(PIO) 8430A(CTC)	同上	同上	同上	同上	¥19,800
GPY-180	HD64180RI	μPD71055C	同上	HM6264ALSP 実装済	同上	可能	¥24,800

GUPPY® SERIESのカタログを用意しております。販売元へご請求下さい。

御注文・お問合わせ
製造元：JCN 日本コムネット(株)
〒541 大阪市中央区南本町3丁目3-23
☎(06)245-7585

●関西地区販売元●
日本コムネット(株)営業部
アドテックインターラブト大阪
〒541 大阪市中央区南本町3丁目3-23-611
☎(06)245-7575(代) FAX.(06)245-7750

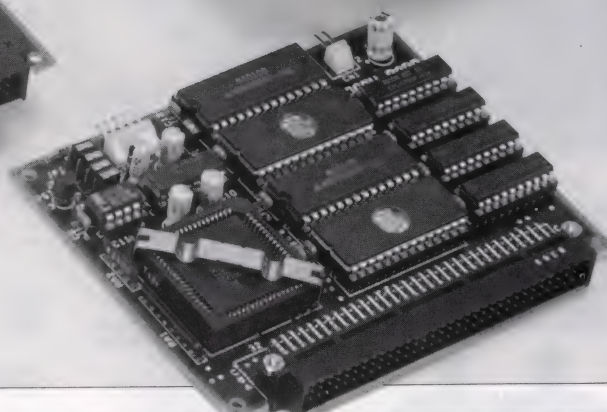
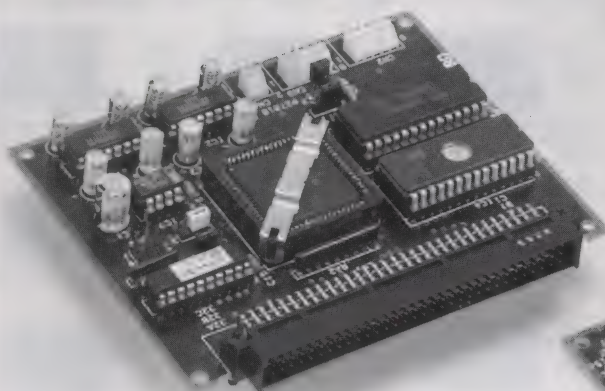
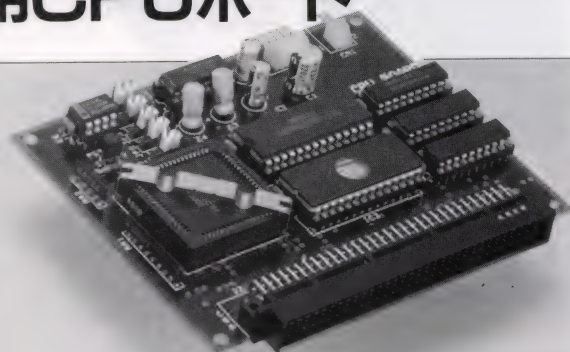
●関東地区販売元●
ソフトマート(株)
〒101 東京都千代田区神田須田町1-18-6
第1谷ビル ☎(03)256-5881(代)

●九州地区販売元●
(株)エフ・エム・イー
〒814-01 福岡市東区多々良1-25-23
たたらグリーンビル ☎(092)662-1561(代)

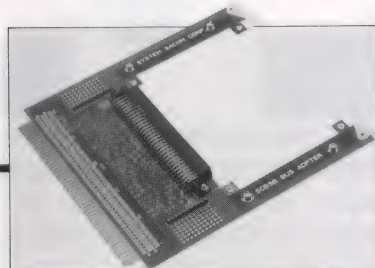
※表示価格には消費税は含まれておりません。

SYSTEM SACOM CORP.
RYOGOKU SAKURAI-BLDG
38-16 4-CHOME RYOGOKU
SUMIDA-KU, TOKYO 130 JAPAN
TEL.03(635)5145 FAX.03(635)5148

B

HD64180、V40、V50使用
製品への組込及び試作用CPUボード
SCB series

SCBシリーズCPUボードはCPU、ROM、RAM、RS-232Cインターフェースにて構成され、CPUの信号線をほとんど外部のコネクターに出していますので製品の試作、製品への組込みなど応用範囲が広がります。
SCBシリーズCPUボードは、V50,V40CPUボードに関しては、外部出力バスは統一してありますので、V40からV50CPUボードへの変更など容易に出来るよう設計されています。またHD64180CPUボードもほぼ統一されていますので、V40,V50CPUボードへの変更も可能となります。
SCBシリーズCPUボードは、小型化、ノイズ関係を考慮し多層基板にて構成されています。

	SCB180-O/SCB180-M*	SCB V40-O/SCB V40-M*	SCB V50-O/SCB V50-M*
CPU	HD64180RICP6	μPD70208L(V40)	μPD70216L(V50)
クロック	6.144MHz	8MHz	8MHz
シリアルインタフェース	2ch	1ch	1ch
	MAX232CPE		
ROM※	27256又は27512相当品×1 ジャンパにてメモリブロック設定	27256又は27512相当品×1	27256又は27512相当品×2
RAM	62256相当品×1	62256相当品×1	62256相当品×2
電源	+5V ±5%		
消費電力	70mA	120mA	200mA
外径寸法	90×115(mm)		
価格	¥24,000(¥29,000)	¥24,000(¥29,000)	¥30,000(¥35,000)

※上記製品にはROMは付属していません　　(*)はモニターROM付です。

※SCB-98バスアダプターは98用スロットバスと完全コンパチブルではありません。一部使えない信号もございます。

開発環境がグーンと拡がります。
SCB-98 BUS
SCB-98バスアダプター ¥20,000

SCBシリーズを、NEC PC-9801シリーズ用スロットバスに変換するためのアダプターです。

HEXTOOL
MS-C, MASM (ver. 4.0, ver. 5.1) ROM化ソフト ¥25,000

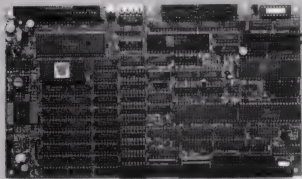
株式会社システムサコム
本社・東京都墨田区両国4-38-16 両国桜井ビル
☎03-635-5145(代)　FAX.03-635-5148
ハードウェア部 直通☎03-635-5417

※MS-Cはマイクロソフト社の登録商標です。
※表示価格には消費税は含まれておりません。

世界で初めて64180を 使った"SPL"の実力!!

B

シングルボードコンピューター
MSC-LAT 1/2

MSC-LATは, 究極の8bitCPU・日立HD64180を業界で始めて採用し, マルチユーザーローカルエリアネットワークをサポートしたシングルボードコンピュータです。

■MSC-LAT1 ················ ¥89,000
■MSC-LAT2(ターミナルタイプ) ········ ¥69,000
■TURBO DOS ················ ¥175,000

●

プログラム開発 システム, 組込機, ローカルエリアネットワークシステムに最高のコストパフォーマンス。

CPU : 64B180 6MHz NON WAIT
RAM : 512KB (標準 : 256KB)
FDD : 3.5/5/8 インターフェース
SCSI : ハードディスクインターフェース
VIDEO : 80×24ch 640×200グラフィック
I/O : RS232C×2セントロニクス

64180高速マクロセンブラSLR180/SLRNK, 高機能通信ソフトchitchat, カラーグラフィックASMライブラリーが標準装備です。
ネットワーク : TURBO DOS (800Kbps)

CP/M PLUS 8bit 最上位のCP/M

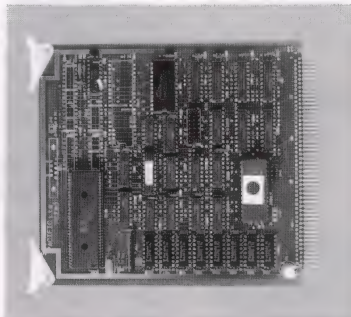
MSC-LATで使われているCP/MPLUSはCP/M2.2とシステムファンクションで上位互換をもつOSです。CP/M86, CP/68Kより上位の設計思想で作られているOSです。

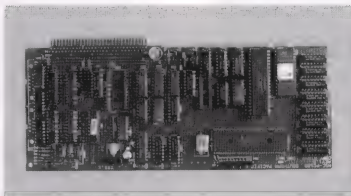
バンクメモリーをサポートしBDOS, BIOS部を裏バンクに置く事により, ユーザエリアTPA60Kを実現しています。ディレクトリーアクセスはディレクトリーバッファをメモリー上に置きハッシュサーチを行ない高速化されています。
ディスク容量の上限も512Mバイト, ファイルサイズは32MバイトとMS-DOSを上廻っています。
ディスクに対してはオートログインをサポートしBDOS ERROR READ ONLY'などは起りません。

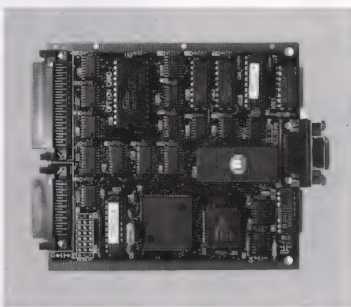
またマルチメディアサポートによりLAT, ICOでは6種類のディスクフォーマットを自動判別。

LAT PCZ80 PC180+TURBO DOS=ZENET!

CP/Mエミュレーター
MSCシリーズ

MSC-PCシリーズも, HD64180を搭載したCP/Mエミュレーターでオペレーティングシステムを意識せずPC9801やJ3100, IBM-PCでCP/M80とMSDOSのプログラムを実行。

PC9801版

■MSC-PCZ80 ················ ¥39,800
■MSC-PCZ80N T.DOS PC付 ······ ¥59,800

IBM-PC版

■MSC-PC180 ················ ¥39,800
■MSC-PC180N T.DOS PC付 ······ ¥59,800

J3100版

■MSC-PCJ80 ················ ¥49,800
■MSC-PCJ80N T.DOS PC付 ······ ¥69,800

IBMPCやJ3100, PC9801で, CP/M80のソフトウェアを実行。
Z80ソフトウェア開発が, MS-DOS上で行なえます。
MS-DOS上でエディト, Z80Hi-techでコンパイル, デバッグ。CP/M80T. PASCALのソースを MS-DOSでエディット, コンパイル, もちろんCP/Mディスクをそのままリードライトも可能。
Z80開発環境を販売してきた実績が生きる。
強力なソフトウェアと高速なハードウェアで快適なプログラミング環境を実現。

〈ソフトウェア〉

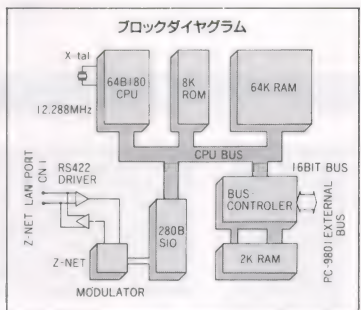
MS-DOSにCP/M shellが常設し, CP/M80のCOM, SUB, MSDOSのCOM, EXE, BATのファイルをオペレーティングシステムを変更することなく実行。MSDOS V3.1の全てのコマンドサーチパスを用いて実行, 16種類のCP/Mフォーマットをドライブに割りあてるシステムドライバ CD.SYS。 RAMディスクドライバRD.SYS|CP/M画面エスケープシーケンスはADM3A上位互換。

※PCZ80 PCZ80Nは旧PC9801, PC98XAではハード, ソフトの違いから動作しません。

〈ハードウェア〉

究極の8bit64180(Z80 7MH相当)搭載, 64Kバイト DRAMをNON WAITアクセス, CP/M TPAは60K相当, メインとのインターフェースはサイクルスチールアクセスによる高速デュアルポートRAM。
コモンメモリ, 64B180リセットはアドレスCøXXX~CEXXXの任意アドレスに変更できる。サポートディスクはNEC5." KAYPRO2. QC10. SPL5"2HD. PCAT5"2HDなど。

〈ローカルエリアネットワーク〉

SPLオリジナルLAN ZENET対応のNバージョン800Kbps, CSMA CD, ツイストペア. TURBO DOS PC+ MSC-LATで高速ネットワーク。

SLR社, HITECH社では当社MSC-LAT, PC180を開発システムとして使っています。

(テ)(ッ)(ク)(サ)(ポ)(ー)(ト) 045-314-1201 PM1:00 - PM5:00
電話の前に, もう一度マニュアルをお読み下さい

Vol.11
★当社取扱いのハード, ソフトそして世界の動きなどの最新情報を満載した情報誌です。是非ご請求下さい。

Ⓙ和文マニュアル Ⓢメディア変換料がかからない商品 ⓂMS-DOS版 ⓅPC-DOS版

SOUTHERN PACIFIC
株式会社 サザンパシフィック

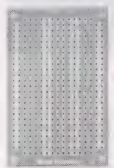
〒220 横浜市西区南幸2丁目16-20 三和横浜ビル3F
TEL 045(314)9514/FAX 045(314)9840
JR横浜駅西口徒歩5分, 営業時間 9:00~17:00 日祝定休

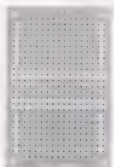
資料請求No.315

Sunhayato ユニバーサル基板

このページに掲載してあるプリント基板はほんの一部です。基板資料・カタログご希望の方は、担当高橋までご請求下さい。

E

ミニシリーズ

ICB-86 ¥110
- 紙ポリエステルHB片面
- 1.6t×47×72
- 孔径1.0φ 2.54ピッチ

ICB-86G ¥170
- ガラスエポキシ片面
- 1.2t×47×72
- 孔径0.9φ 2.54ピッチ

ICB-88 ¥110
- 紙ポリエステルHB片面
- 1.6t×47×72
- 孔径1.0φ 2.54ピッチ

ICB-88G ¥170
- ガラスエポキシ片面
- 1.2t×47×72
- 孔径0.9φ 2.54ピッチ

ICB-87 ¥110
- 紙ポリエステルHB片面
- 1.6t×47×72
- 孔径1.0φ 2.54ピッチ

ICB-89 ¥140
- 紙ポリエステルVO片面
- 1.6t×47×72
- 孔径1.0φ 2.54ピッチ

ICB-90 ¥195
- 紙ポリエステルHB
- 孔径1.0t 2.54ピッチ

96シリーズ

ICB-96 ¥670
- 紙ポリエステルHB片面
- 1.6t×115×160
- 孔経0.9φ 2.54ピッチ

ICB-96G ¥750
- ガラスエポキシ片面
- 1.2t×115×160
- 孔径0.9φ 2.54ピッチ

ICB-96GHD ¥3,550
- ガラスエポキシ両面 スルホール仕上
- 1.6t×115×160
- 孔径0.9φ 2.54ピッチ

ICB-96PU ¥670
- 紙ポリエステルHB片面
- 1.6t×115×160
- 孔径0.9φ 2.54ピッチ

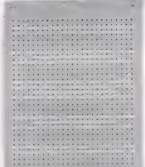
ICB-96GU ¥880
- ガラスエポキシ片面
- 1.2t×115×160
- 孔径0.9φ 2.54ピッチ

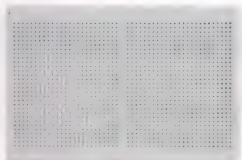
ICB-96P2 ¥670
- 紙ポリエステルHB片面
- 1.6t×115×60
- 孔径0.9φ 2.54ピッチ

ICB-96GHD ¥3,550
- ガラスエポキシ両面 スルホール仕上
- 1.6t×115×160
- 孔径0.9φ 2.54ピッチ

93シリーズ

ICB-93S ¥330
- 紙ポリエステルVO片面
- 1.6t×72×95
- 孔経1.0φ 2.54ピッチ

ICB-93SG ¥380
- ガラスエポキシ片面
- 1.2t×72×95
- 孔径1.0φ 2.54ピッチ

ICB-93SGH ¥980
- ガラスエポキシ両面 全孔スルホール
- 1.6t×72×95
- 孔径0.9φ 2.54ピッチ

ICB-93S-2 ¥330
- 紙ポリエステルVO片面
- 1.6t×72×95
- 孔径1.0φ 2.54ピッチ

ICB-93W ¥565
- 紙ポリエステルVO片面
- 1.6t×138×95
- 孔径1.0φ 2.54ピッチ

ICB-93WGH ¥1,760
- ガラスエポキシ両面 全孔スルホール仕上
- 1.6t×138×95
- 孔径0.9φ 2.54ピッチ

140シリーズ

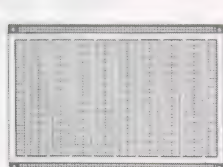
ICB-141G ¥3,000
- ガラスエポキシ片面
- 1.6t×130×180
- 孔径0.9φ 2.54ピッチ

ICB-141GHD ¥4,200
- ガラスエポキシ両面 スルホール
- 1.6t×130×180
- 孔径0.9φ 2.54ピッチ

ICB-141GU ¥3,000
- ガラスエポキシ片面
- 1.6t×130×180
- 孔径0.9φ 2.54ピッチ

ICB-141GHU ¥4,200
- ガラスエポキシ両面 スルホール仕上
- 1.6t×130×180
- 孔径0.9φ 2.54ピッチ

130シリーズ

CPC-131 ¥1,800
- ガラスエポキシ片面
- 1.6t×96×150
- 孔径0.9φ 2.54ピッチ

CPU-132 ¥3,500
- ガラスエポキシ片面
- 1.6t×198×137
- 孔径0.9φ 2.54ピッチ

CPU-133 ¥4,500
- ガラスエポキシ片面
- 1.6t×156×230
- 孔径0.9φ 2.54ピッチ

98シリーズ

ICB-98 ¥1,750
- 紙ポリエステルVO片面
- 1.6t×137×232
- 孔径0.9φ 2.54ピッチ

ICB-98G ¥3,950
- ガラスエポキシ片面
- 1.6t×137×232
- 孔径0.9φ 2.54ピッチ

ICB-98SEG ¥4,700
- ガラスエポキシ両面
- 1.6t×137×232
- 孔経0.9φ 2.54ピッチ
（パーツサイド・シールドパターン）

ICB-98GH ¥5,500
- ガラスエポキシ両面 全孔スルホール仕入
- 1.6t×137×232
- 孔径0.9φ 2.54ピッチ

ICB-96GU ¥3,950
- ガラスエポキシ片面
- 1.6t×137×232
- 孔径0.9φ 2.54ピッチ

ICB-98GHU ¥5,500
- ガラスエポキシ両面 全孔スルホール仕上
- 1.6t×137×232
- 孔径0.9φ 2.54ピッチ

【98シリーズはこの他にも沢山あります。】

●製品の外観および仕様は改良のため変更することがあります。
●この誌面に掲載の全商品の価格には消費税は含まれておりません。ご購入の際、消費税が付加されますのでご承知おき願います。

サンハヤト株式会社 本社 〒170 東京都豊島区南大塚3-40-1 ☎03(984)7791代 FAX 03(971)0535

Sunhayato 新製品情報

このページの資料請求は担当虻川まで

オゾン層保護　ノーフロンスプレー開発に成功

不燃性・無公害・安全

●本剤は炭酸ガスをフロン代替噴射剤として活用した画期的な新製品です。

●バルブ本体は、半永久的にご使用いただけます。

●消耗品のガスボンベは高圧ガス取締条例による最大量限度一杯の95mℓ入りで、二重安全弁付です。

●特に急冷剤は、従来のフロンスプレーに比べて、少量噴射で急速な冷却効果があります。

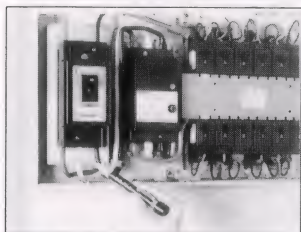
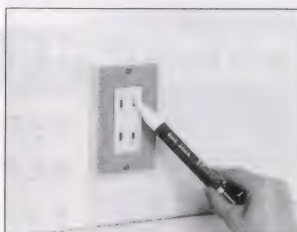
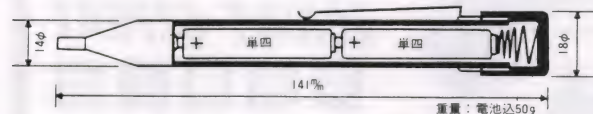
品　名	註	内　訳	型　名	標準価格	備　考
急冷剤 CO_2 QREI タンサンキューレイ	Ⓐ	バルブ本体	CO2-01	18,800	半永久備品
	Ⓑ	急冷剤ボンベ	CO2-9	1,850	消耗品 95mℓ
		Ⓐ+Ⓑ急冷剤セット	CO2-901	20,650	
ほこり吹き飛ばし用 CO_2 BLOW タンサンブロー	Ⓐ	バルブ本体	CO2-01	急冷剤用と同じ	
	Ⓑ	ブローボンベ	CO2-8	1,650	消耗品 95mℓ
		Ⓐ+Ⓒブローセット	CO2-801	20,450	

●出張サービスなどに便利な携帯用ハードケースを近日発売予定です。

裸線に触れずに通電検知!!

■配電盤の通電チェック

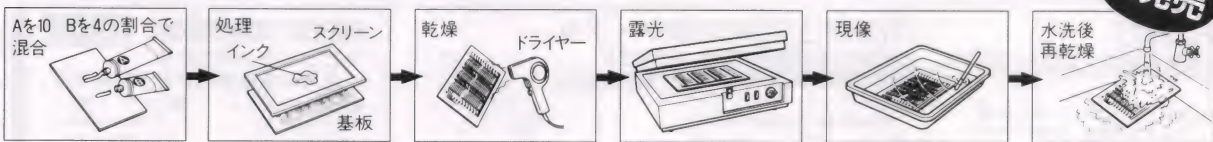
■電庭・会社などの通電チェック

新発売

AC Volt Stick

PAT. P.

MODEL　VS-621 ¥4,800

●埋込ソケット・分電器・配電盤内の通電チェック
●ケーブルの断線個所検知
●プラグ及びヒューズホルダー内のヒューズ切れ検知
●単相/三相用ケーブルの通電チェック
●不良インラインスイッチのチェック
●安全器の検査
●クリスマスデコレーション電球などの直列配線の電球切れ位置のチェック

フォトソルダー（半田）レジストセット

新発売

●チューブ入り絵具状のレジストで、紫外線露光・アルカリ現像です。
●すきとおったグリーンの被膜が均一に塗付され、専門工場の製品と同じように仕上ります。
●この被膜は基板によく密着し、耐久性、電気絶縁性に優れ、手ハンダ又は自動ハンダ槽に耐えます。

※ソルダーレジストは冷暗所保存のため、受注発送になります。また、その際にクール宅配便を利用しますので送料がかかります。

品　名		容　量	数量	型　名	セット型名	標準価格
ソルダーレジスト	A剤	70g	1本	SR-321	SR-320	¥3,300
	B剤	30g	1本	SR-322		
現像剤		100g	2袋	SR-323		
ポリ手袋			1組			
スクリーン		216×306mm	1枚	SR-341	SR-340	¥4,800
スキージ		200mm	1ヶ	SR-342		

●製品の外観および仕様は改良のため変更することがあります。
●この誌面に掲載の全商品の価格には消費税は含まれておりません。ご購入の際、消費税が付加されますのでご承知おき願います。

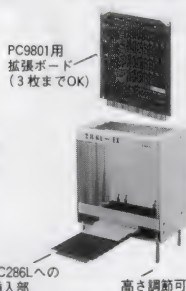
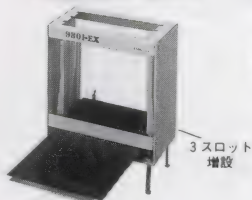
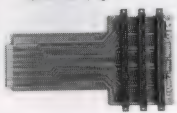
サンハヤト株式会社　本社 〒170 東京都豊島区南大塚3-40-1
☎03(984)7791(代) FAX 03(971)0535

ケーブル・切換BOX

★ケーブルの長さは特注に応じます。

拡張スロットボックス

NECPC9801用
9801-EX

エプソンPC 286L用
286L-EX

●ボード単体の販売も致します。

■PC9801用拡張ボードが3枚まで搭載できます。
■外部電源入力用ターミナルと、切替えスイッチがあります。

セントロニクス(36P) RS232C(25P) 切換BOX

(セントロニクス(プリンター) 36PIN全接点切換BOX)

セントロニクス
■CB-13(1コモン3ch切換)
■CB-14(1コモン4ch切換)

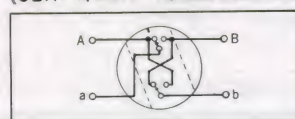
(RS232C25PIN 全接点切換BOX)

RS232C
■DB-13(1コモン3ch切換)
■DB-14(1コモン4ch切換)

〈セントロニクス〉
- CB-12 ¥ 8,800
- CB-13 ¥11,000
- CB-14 ¥12,000

〈D-SUB〉
- DB-12 ¥ 8,000
- DB-13 ¥ 9,000
- DB-14 ¥11,000

クロス切換器

《CBX-4、DBX-4の回路ブロック図》

NEC9801/8801用プリンターケーブル

C1436M-S1

- セントロ準拠14P-36P
- アルミフォイルシールド線
  - 2m ¥1,900
  - 3m ¥2,200
  - 5m ¥2,750

36芯セントロ延長ケーブル

C36M-S4

- 36P-36P
- 全ピン1対1接続
- ツイストペア線で高耐ノイズ
- 編組シールドケーブル
  - 2m ¥2,850
  - 3m ¥3,450
  - 5m ¥4,500

98用PLDユニバーサルプログラマー

- EPROM
- PLD
- EPLD
- GAL
- PROM
- ワンチップマイコン

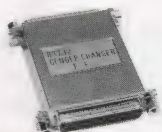
――をサポート!

SPRINT-EXPERT

実演展示中

PROMAC製

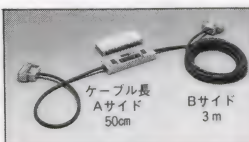
RS232C GENDER CHANGER F/F

接続したいコネクター同士が両方共オスの場合、ご使用下さい。簡単に接続できます。

RS232C用接続切換ケーブル

RS-10Aシリーズ

★EMI対策品
(電磁障害防止品)

ラインナップ!

■RS232Cインターフェイスを持つ機器間の相互接続が制御信号線の違いに関係なく、パネルのDIPスイッチをON-OFFすることで接続できます。

■RS10AX…25P－25P
コネクタケーブル用(汎用)
■RS10AJ…9P－25P
コネクタケーブル用

ホスト(東芝)J3100用

■RS10AK…15P－25P
コネクタケーブル用

RS232C結線切替器 RS-15

★ピンボード式だから
あらゆる結線組合せが可能です。

■制御線5本のリターン用DIPスイッチがAサイドとBサイド両方にあり、両サイド分離でリターン回路ができます。

■ピンボード用ショートピンを使えばシンクロスコープ等の測定器により波形観測が簡単にできます。

チェッカーとして最適!

ESEC 日本エセック株式会社

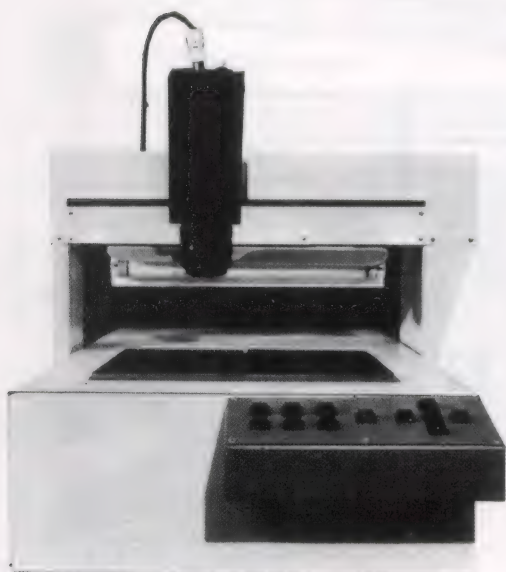
本　　社/〒160 東京都新宿区西新宿7-18-13ハイム大成ビル TEL.03(369)8061(代) FAX.03(365)3005
千葉事業所/〒270-11 千葉県我孫子市布佐平和台3-14-20 TEL.0471(89)4632 FAX.0471(89)4712

※9月より本社を下記へ移転致します。
〒101 東京都千代田区外神田2－10－8
富士ビル7F
TEL.03(5256)3166(代) FAX.03(5256)3167

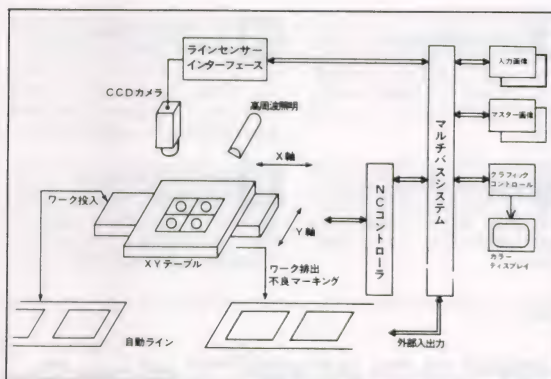
資料請求No.318

クボテック計測検査システム

光学式外観検査装置

検査物を移動させながらCCDラインセンサーからの信号を入力し、
二次元画像としてメモリー上に取り込み、各種検査を行うものです。
基準となるマスター画像と比較し、異なった部分が即座に検出できます。

〔特 長〕

- 4000画素×8000画素の高分解能のため、微小欠陥が検出できます。
- 入力フレーム・メモリは２画面あり、画像取り込み中に他の画面を同時に処理できるため、処理時間を短縮できます。
- ２値化のしきい値設定は、CCD１ライン分の補正メモリにより１画素ごとに行われます。
- 画像処理専用ビットスライス・プロセッサにより高速処理します。
- パターン・マッチング後の微小ノイズは各種フィルタにより除去できます。
- 欠陥部の位置と画像が表示されます。
- 位置決めマークを合わせることにより、検査物の位置ずれを補正することができます。
- ＮＣ装置により、一枚のワーク上の同一パターンを複数個検査できます。
- マスター・パターンをティーチングし、位置決めマーク位置をマウスで登録するだけで検査データが作成できます。
- 検査データはオプションのハード・ディスクに保存することができ、品種の切り替えが簡単にできます。

〔構 成〕

〔仕 様〕

入力方式	：CCDラインセンサ (5000画素) ディジタルカメラ (8ビット)
水平スキャン時間	：1.0ms~2.0ms
入力用フレームメモリ	：4000×8000×2 画面
グラフィック・コントローラ	：日立 HD63484 (ACRTC)
画面表示	：1280×1120 16色
CPU	：MC 68000
ビットスライス・プロセッサ	：Am29117 他

自動化を推進する――――

KUBOTEK クボテック株式会社

〒530 大阪市北区中之島4-3-36 玉江橋ビル ☎〈06〉443-1815. FAX〈06〉443-6039
〈東京営業所〉〒103 東京都中央区日本橋富沢町12－18 ☎〈03〉664-3573

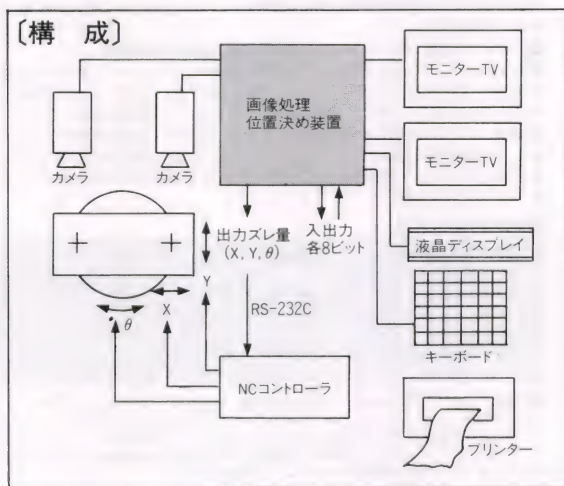
クボテック画像処理システム

画像処理位置決め装置

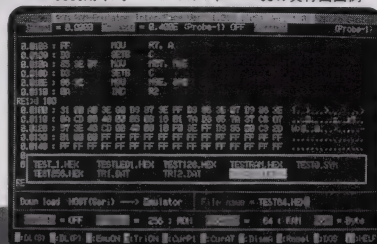
2台のCCD・TVカメラより入力された画像から、基準マークのズレを測定し、
ワーク全体をX、Y、θの3軸で補正します。

■特　長

- テレビカメラの視野、マークの位置及び機械系の各種のパラメータがキー入力可能なために広い範囲に応用できます。
- 出力はX、Y、θ軸のパルス単位で出力されるため容易にNC装置と接続できます。
- 基準位置がティーチングできるため、位置合わせが簡単です。
- 基準マークとして十字、円のほかプログラムによって対象ワークの特徴点(例えばICのピン、コーナー等)が使用できます。
- RS-232Cインタフェースを備えており外部との接続が容易です。
- プリンターとの接続ができ、データのプリントアウト等が可能です。

H

■仕　様

- テレビカメラ：CCD、外部同期方式
  最大4台接続可
- モニターTV：5インチ2台、原画像・メモリー画像切換可
- 画像メモリー：512×480画素×1ビット
  4画面分
- データ入力：キーボード及び液晶ディスプレイ
- 入出力：フォトアイソレーション
  入出力各8ビット
  RS-232C 2チャンネル
  プリンターインタフェース
  セントロニクス仕様

自動化を推進する
KUBOTEK クボテック株式会社
〒530 大阪市北区中之島4-3-36 玉江橋ビル ☎(06)443-1815
FAX.(06)443-6039

RRE-1000は、EPROMやスタティックRAMなどのROMおよびRAMの、プログラムやデータ等をエミュレートするための装置です。ROMのエミュレートだけでなく、RAMのエミュレートも可能にしたことで、ターゲットボードのCPUがRAMにどのようなデータを書き込んでいるか、またスタックエリアをどのくらい消費しているかなどを見ることができます。更に、カレントアドレス表示機能、トリガー機能、メモリーバックアップ機能など従来のROMエミュレーターにはない機能を持っており、まさに、ROM化する開発現場で生まれたROM/RAMエミュレーターです。

■ROM、RAMに対応。

RRE-1000はROMだけでなく、RAMもエミュレート可能です。これにより、バグ解析等にたいへん役立ちます。
(64Kビット、256KビットのRAM対応)

■ホストマシンに限定されない。

ホストのパソコンはMS-DOSマシンでAUX(シリアル)の使用可能なマシンであれば機種を問いません(RS232のコネクターは25ピンD-SUBタイプ)。PRN(プリンター)のコネクターも、RRE-1000と付属ケーブルで接続しておけば、高速転送が可能です。また、インテルHEXフォーマットや、RRE-1000のコマンドを記述したコマンドファイルであれば、MS-DOSのコマンドラインからcopyコマンドでRRE-1000に、直接データを転送することができます。

■PC-9800用のインターフェースソフトが付属。

PC-9800シリーズ用(注1)にウィンドウ型インターフェースソフト・RRE-98が付属しています。このRRE-98は、RRE-1000のROM/RAMがアクセスされているアドレスのオート表示、バイナリーファイルの転送、逆アセンブル(注2)、シンボリック表示、コマンドのヒストリー、ファイルウィンドウなど高機能で且つ、操作性重視のインターフェース・プログラムです。

*注1:PC9800シリーズのハイレゾモードを除く。IBM-PC版開発中
*注2:Z80(64180)、インテル8051/85/86/96/196対応

●広告に記載されている内容は、製品改良のため予告なく変更される場合がありますのでご了承下さい。
●MS-DOSは米マイクロソフト社の登録商標です。●Z80は米ザイログ社の登録商標です。

■ROM/RAM・エミュレーターRRE-1000の主な特徴

●64Kビットから1024Kビットまで対応●ROM/RAMのエミュレート可能●カレントアドレス機能●トリガーアドレスブレーク機能●トリガーアドレスパルス出力機能●リセットコントロール出力機能●バイト/ワードモード対応●エミュレート用RAMのバックアップ●各種モードをソフトで変更可能●各種モードをEEPROMに記憶●最高4台まで接続可能

■ウィンドウ型インターフェースソフトRRE-98の主な特徴

●カレントアドレスのオート表示機能●バイナリーファイルを含むファイルのダウン/アップロード機能●ファイルウィンドウ表示スクロールアップ/ダウン機能●ファイルセーブ機能●逆アセンブル機能●シンボル表示●画面のカラーカスタマイズ機能●ファンクションキー使用可能●ダイヤモンドカーソルキー使用可能●コマンド入力のヒストリー機能●MS-DOSコマンドの実行●MS-DOSのチャイルドプロセス実行

■製品構成と価格

製品	価格	構成
RRE-1000モデル1	¥98,000	●64Kビット×2~256Kビット×2 ●ICソケートプローブ28P×2
RRE-1000モデル2	¥102,000	●64Kビット×2~512Kビット×2 ●ICソケットプローブ28P×2
RRE-1000モデル3	¥104,000	●64Kビット×2~512Kビット×2 ●1Mビット×1 ●ICソケットプローブ 28P×2、32P×1(1MビットROM専用)

●付属品 ACアダプター(DC9V300mA出力)/RS-232C用ケーブル(2.5m)/パラレル用ケーブル(2.5m)/リセットコントロール用クリップ(30cm)/各モデル別ICソケットプローブ(30cm)/取扱説明書2冊(RRE-1000, RRE-98)/ユーザー登録カード/RRE-98ディスケット(逆アセンブルのCPUの種類は1種類のみ付属。MS-DOSは組み込まれていません。)
●オプションケーブル(増設用ケーブル) ¥1,000(1本)
RS-232C用ケーブル(30cm)/パラレル用ケーブル(30cm)
●オプションソフト ¥5,000(1CPU)
各CPU別逆アセンブル機能付RRE-98

表示価格には消費税は含まれていません。

販売代理店募集

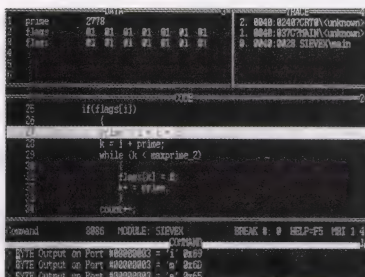
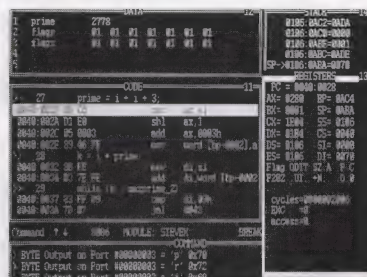
Micro System Co., Ltd.
開発、製造(有)マイクロシステム
住所 〒791-02 愛媛県松山市水泥町333-282
TEL(0899)76-7669 FAX(0899)76-4490

これが〝時代の選択〟だ！

《マイクロテックがおとどけします》

ハイパフォーマンス"C"言語シュミレートデバッガ

XRAY

ラインナップ
- XRAY68K
- XRAY86
- XRAY Z80
- XRAY G32

Cソースコードレベルスクリーン

アセンブラレベルスクリーン

走行環境

開発対象 MPU	MCC CROSSC COMPILER	ASM CROSS ASSEMBLER	XRAY シュミレータ バージョン	XRAY ICE バージョン
68020/ 68000/68010	MCC 68K	ASM 68K	XRAY 68K	C-ICE 68020 C-ICE 68000/10
80286/ 80186/80188 8086 8088	MCC 86	ASM 86	XRAY 86	C-ICE 80286(リアル) C-ICE 80186/188 C-ICE 8086/88
Z80 HD64180	MCC Z80 MCC 180	ASM Z80 ASM 180	XRAY Z80* XRAY 180*	C-ICE Z80 C-ICE 64180
Gmicro/200	MCC G32	ASM G32	XRAY G32	
100	※	※	※	
300	※	※	※	

Gmicro/100、300は現在開発中です＊印は開発中です。
VAX8000シリーズ(VMS)、Macro VAX11/Macro VAX2000シリーズ(Macro VMS)、SUN-3(UNIX)、東芝AS-3000(UNIX)、APOLLO(DOMAIN IX)、HP-9000/3300(UNIX)、ソニーNEWS(UNIX)、IBM PC/AT(PC-DOS)、東芝J-3100(MS-DOS)、NEC PC-9800シリーズ(MS-DOS)、Data General ONE(MS-DOS)、シャープMZ-2861、その他

※XRAYは米国マイクロテックリサーチ社の登録商標です。
※VAX、VMS、ULTRIXはDEC社の、IBM PC/ATはIBM社の登録商標です。UNIXは米国ベル研の開発したOSです。
※記載事項はおことわりなしに変更することがあります。

高信頼性インサーキットエミュレータ

C-ICE

CORE **in-circuit emulator**

対応CPU
- インテル 80386/80286/80186/8086/8085/8051ファミリー
- モトローラ 68020/68010/68000/68HC11/6809
- その他

販売特約店

マイクロテック株式会社
本　社 〒160 東京都新宿区西新宿7-9-17 伊藤ビル ☎03-371-1811
厚木営業所 〒259-11 神奈川県伊勢原市下落合626-16 ☎0463-96-2301
大阪営業所 〒541 大阪市中央区道修町2-2-5 三忠ビル ☎06-227-1288
名古屋営業所 〒464 名古屋市千種区東山通5-30-2 中野殖産ビル ☎052-782-1603

L

新・登・場
パソコン用
光磁気[MO]ディスクシステム

JP2020は、大量の情報処理を可能にする光磁気ディスク(Magneto Optical Disk)ユニットです。メディアは5.25インチのカートリッジを使用し、片面297MB、両面594MBの大容量に加え、消去、書込み、ランダムアクセス可能、リムーバブルといった従来のデバイスにはない特徴を持っています。大量の文書や画像データの保管、交換、配布、バックアップ等に有効です。

JP2020

■仕様

使用メディア	カートリッジ型130mm(5.25inch)MOディスク
容量	両面:594MB、片面:297MB
インターフェース	SCSI
フォーマット	31 Sectors/track 512B/Sector 18751Tracks/Side
回転速度	2400rpm
回転モード	CAV
平均回転待ち時間	12.5msec
シークタイム	22msec(±64track以内) 95msec(平均)
データ転送速度	620KB/sec(ノーエラー時、連続) 1.2MB/sec(バースト)
外形寸法	126(W)×310(D)×211(H)mm
重量	6.5kg

■適応パソコン

型名	適応機種
JP2020/P	NEC PC-9801、XL、EPSON PC-286シリーズ
JP2020/F	富士通 FMRシリーズ
JP2020/T	東芝 J-3100シリーズ
JP2020/55	IBM PS/55シリーズ
JP2020/70	IBM PS/2、5550S/T、5570Tシリーズ
JP2020/51	IBM PC-AT、各社AX機シリーズ

充実のLINE UP
パソコン用
磁気テープシステム

各研究機関より提供されるさまざまな研究データ、民間企業より提供される各種データベース等、MT(磁気テープ)を標準メディアとするものは今後も増大すると伝えられています。JXシリーズは、今までミニコンや汎用機でしか利用できなかったMTをパソコンでも簡単にしかも低価格で導入することを可能にしたパソコンのためのMTシステムです。

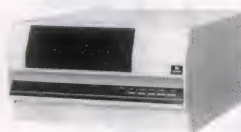
■MTによるデータ提供例

●天体観測データ ●気象データ(アメダス) ●各衛星データ ●地震データ ●NCデータ ●自動車関連CADデータ ●画像データ ●CGデータ

JX8420

■仕様

記録方式/密度	800/1600/3200BPI (NRZI/PE/DDPE)
テープ速度	100IPS(ストリーミングモード) 45IPS(スタートストップモード)
データ転送速度	36KBPS~1MBPS
巻き戻し速度	160IPS (2400ftテープの場合約3分)
使用テープ	JIS C6240、幅½インチ、長さ2400ft (リールサイズ:6、7、8、10号)
テープバッファ方式	テンションアーム方式
バッファメモリ	128KB
外形寸法・重量	272.5(H)×480(W)×656(D)mm・44kg (JX8410) 200(H)×480(W)×550(D)mm・34kg (JX8420)
環境条件(動作時)	周囲温度:5~38°C 相対湿度:20~80%

JX5320

■仕様

記録方式/密度	1600/6250BPI (PE/GCR)
テープ速度	100IPS
データ転送速度	20KBPS~1MBPS
巻き戻し速度	160IPS (2400ftテープの場合約3分)
使用テープ	JIS C6240、幅½インチ、長さ2400ft (リールサイズ:7、8、10号)
テープバッファ方式	リール間直接駆動(テンションアーム併用)
バッファメモリ	512KB
外形寸法・重量	347(H)×482(W)×656(D)mm・60kg (JX5310) 274.5(H)×482(W)×656(D)mm・50kg (JX5320)
環境条件(動作時)	周囲温度:5~35°C 相対湿度:20~80%

■JX8420/JX5320 適応パソコン

PC-9801シリーズ、PC-286シリーズ、FMR、IBM 5550、IBM PC/AT、PS/2 他

《中古パソコン情報》

▶NEC PC-9801VX2・VX21・VX41・UX21・UX41・LV21

★上記機種の中古パソコン在庫あります。
詳しくは渋谷営業所TEL.03-496-3183までお問合せください。

◎関東地区でのご用命は渋谷営業所までお願い致します。 ※資料請求及びお問い合わせは、弊社システム・サポートI/F係宛へ ※製品価格には、消費税は含まれておりません。

〒556 大阪市浪速区日本橋5-11-14 丸誠第2ビル2階 TEL.06-644-6446代 FAX.06-644-0070
渋谷営業所:〒150 東京都渋谷区桜丘町29-35 TEL.03-496-3183 FAX.03-496-3860

M

資料請求No.325

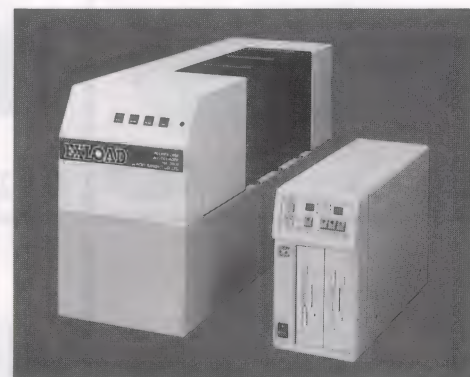
RICOH
RICOM
NPR5200
テープリーダ/パーフォレータ
★高速読取，高速さん孔が可能です。
★占有面積を最少限に押えるように設計してあります。
★受信メモリ内蔵により通信効率を高めることができます。
★ISO↔ASCII，EIA↔ASCIIのコードコンバートができます。
★RS-232Cを標準装備しております。
RS-232Cインターフェイス
ライン/コード、コントロール方式、ストップビット数、
ボーレート切替は可能です。
外寸
280(W)×275(H)×320(D)㎜
OEM
豊富な納入実績を誇る
紙テープリーダ/パーフォレータを
ご利用ください。
リコー電子工業株式会社
〒104 東京都中央区銀座6-13-7☎03(545)3791(大代表)

M

FLOPPY DISK DUPLICATOR

新製品

ONE to ONE® シリーズ

信頼と実績のデュプリケーター

●**簡単操作**　フォーマット自動解析機能(プロテクトには対応しません。)により、マスターディスクのフォーマットを自動解析し、コピーを行ないますので、複雑なオペレーションのいらないスイッチポンの簡単操作を実現しました。

●**ハイスピード**　FDC(フロッピーディスクドライブ・コントロールLSI)を効率的に使用することにより、無駄な回転のない高速コピーを実現しました。しかも、オプションの増設メモリーユニットを付加することにより、メモリー to ディスクコピー方式が可能となり、2ドライブ同時処理で処理能力は倍増されます。

●**正確に**　コピー後のベリファイ処理により書かれたデータは確実に保証されます。

●**全自動コピー**　フロッピーディスク・オートローダーEXLOADシリーズとの接続でフロッピーディスクドライブへの搬入、搬出が自動化され全自動コピーが実現されます。

豊富なラインナップ
"One to One"シリーズは、5.25"タイプの他に3.5"タイプおよびコンバートタイプを揃えております。

※本機は著作権法で許された範囲のコピーを目的として使用するものです。

M

報映産業株式会社　情報機材部

〒103 東京都中央区日本橋小舟町12-15日本橋ビル TEL 03 (665)3430
〒550 大阪市西区新町1-10-2大阪産業ビル TEL 06 (532)3361
〒460 名古屋市中区丸ノ内3-15-26 TEL 052(951)6031
〒810 福岡市博多区中洲5-6-20明治生命ビル TEL 092(271)5281
〒730 広島市中区鉄砲町1-20第3ウエノヤビル TEL 082(223)7779
〒980 仙台市国分町1-6-18東北王子不動産ビル TEL 0222(24)1860

TECHNO PARK 峯

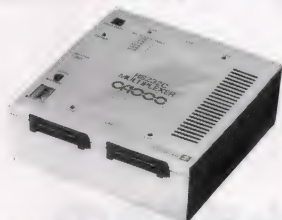
4対12にスクランブル結線！

WANTED

RS-232C
4：12スイッチセレクター
M-412

■4対12のスクランブル結線が行え、各チャネルについて12通りの接続が可能。
■各チャネルは全てDCEモード。

ズラリ揃ったRS-232C！

RS-232Cマルチプレクサ3兄弟！

M-100A・M-100A MarkII・α1000

多チャンネルを網羅！

RS-232C
マルチプレクサ
M-1032

α-1000と
myLinkでスクラム

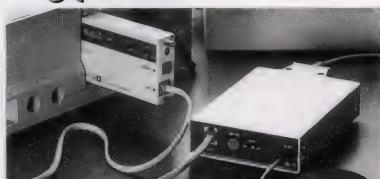
RS-485
コンセントレータ

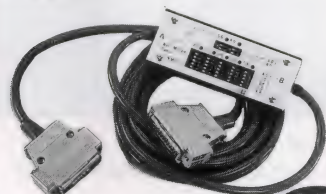
RS-232Cをスクランブル結線。

4:4スイッチセレクター
M-404

RS-232Cをスクランブル結線。

4：12スイッチセレクター
M-412

Mylink
ローコストでシンプルなLANに最適。

マルチポイントモデム
Model 10・11

これは便利！

へんしんケーブル®
A-100
NMC-2S

ラインバッファ兼多機能スピードコンバータ。

スピードコンバータ
BF-5963

開発中！

プロッタ・バッファ
MPB-200

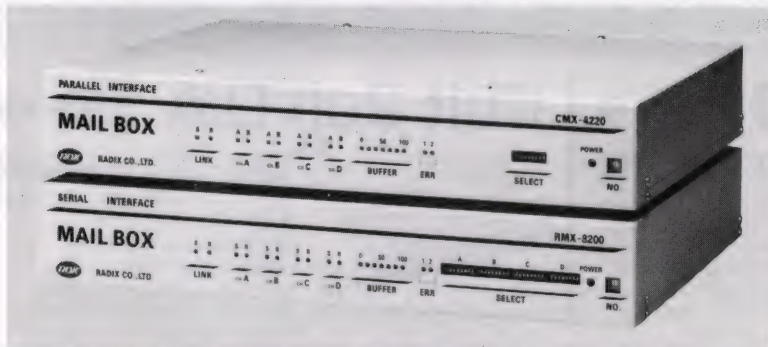
●コンピュータ応用システムのソフトウェア及びハードウェア開発も承っております。

●フロッピーコピー業務も承ります。

●明日の技術に挑戦する
株式会社 テクノパーク 峯

本社：東京都新宿区新宿2-1-15 ふるたかビル7F ☎03(350)5036 担当：加藤

資料請求No.328

Radix creates your Communication world.

"THE MAIL BOX"

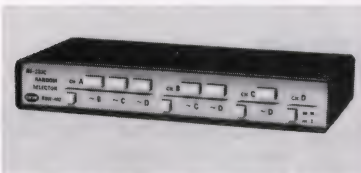
PARALLEL INTERFACE
CMX-4220
(4ch.) ¥265,000

SERIAL INTERFACE
RMX-8200
(4ch.) ¥265,000

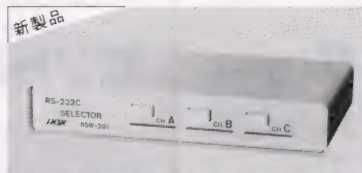
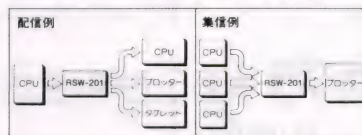
コマンド操作プログラム
(BASIC記述)
多回線BBSプログラム
(BASIC記述)

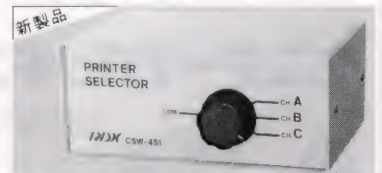
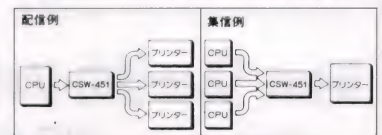
(BBS開局)
電話番号 03(866)6110
接続条件 N81XN
300/1200全二重
ゲストID GUEST

THE MAIL BOXは多機能の通信制御装置です。メーカを超え機種を超えあらゆる計算機、パソコン、周辺装置、計測機器を結合できます。

[構成] THE MAIL BOXは1台当り4チャネル＋ネットワーク制御。RS-232C仕様のRMX-8200とセントロ仕様のCMX-4220の2機種があります。各BOXごとに256K(1MB拡張可)のメモリを装備。
[ネットワーク] THE MAIL BOXは1台の単独使用から最大16台、64チャンネルまで拡張できます。しかも全チャンネル対称ですので特にネットワーク制御装置を付加する必要はありません。BOX間結合には128Kボーの高速ループネットワーク方法を採用。コンテンション発生時のトラブルはありません。
[制御] THE MAIL BOXはコマンド制御。基本コマンド22種類、補助コマンドを含めると70種類を越えるコマンドがプログラミングの手間をはぶきます。さらにコマンドのチャンネル間転送機能により他チャンネルの遠隔制御やモニタリングができます。
[通信方式] THE MAIL BOXのチャンネル間通信はパケット交換によるメイル通信方式。相互通信、同時受信、同報通信、マルチドロップはもちろん、大容量メモリを標準装備していますので受信チャンネルが作業中でもデータ転送できます。
[用途] データ収集、端末制御、伝票配送システム、公衆回線制御、群管理制御、BBS、スプーラ、インターフェイス変換、一時記憶等………
★用途は無限。ぜひ一度御相談下さい。

パソコン間のデータ交換はRSW-402で。最大4台のパソコンのRS-232Cインターフェイスを釦1つで任意に接続できます。6信号線独立切換えです。

RS-232C RANDOM SELECTOR
RSW-402 (4ch.) ¥56,000

新製品

周辺装置の増設や共用は、RSW-201で。1対3、3対1のいずれの分配にも使えます。8信号線を独立切換え。プッシュボタンで切換えができます。

RS-232C SELECTOR
RSW-201 (3ch.) ¥31,000

新製品

用紙の掛換えの手間を省くにはCSW-451で。ツマミ1つで切換えられます。セントロ接続の21信号線を独立切換え。プリンタの共同利用にも使えます。

PRINTER SELECTOR
CSW-451 (3ch.) ¥33,000

■使用可能機種PC-9801/FM11/J-3100/パソピア16/UX-300/MB16007A/if-800/NCR9005/MZ3500/TALK570/C-18/SORDM23, M68/AppleII, III/intelMDS/IDS8200/IBM5550/IBM5150/…他

RDX
株式会社ラデックス

●本　　　社 東京都千代田区岩本町2-10-9タイショウビル 〒101 TEL.03(864)8021(代表)
●名古屋営業所 名古屋市中区錦2-20-20大和生命ビル 〒460 TEL.052(231)1721
●㈱システック福岡 福岡市中央区天神4-9-12第二正友ビル 〒810 TEL.092(752)1234
●㈱システック沖縄 沖縄県那覇市前島2-21-8ふぞうビル 〒900 TEL.0988(62)9900
●製品のお問合せは03(864)8021

M

M

N

宇宙でも使われているSmalltalk/V

ジョージア工科大学のクリスチーン・ミッチェル博士はNASAから委託されたプロジェクトで，新しいマン・マシン・インターフェースの不可欠な部分として，3大AI言語と言われているSmalltalk/Vを採用することに決めました。
Smalltalk言語で書かれたアプリケーションが常に衛星通信のオペレータへの指令やネットワークの状態と，プロジェクト全体を制御しています。
NASAでSmalltalk/Vと言えば，リアルタイム，本物のオブジェクト指向プログラミング(OOPS)実用をあらわしています。

仕様

機種：IBM PC/XT/AT,J3100.PC-DOS
ドライブ：フロッピ2台もしくは，ハードディスク1台
RAM：512K以上
表示：CGA.EGA.Hercules.ATT
640×200dot白黒
フローティングポイント演算には，要8087
マイクロソフトマウス使用可能
機種：PC9801 フロッピorハードディスク
RAM640Kカラーor白黒 640×400
NECバスマウス使用可能

Smalltalk/V……………………Ⓙ[M]¥49,800Ⓢ
……………………Ⓙ[P]¥39,800Ⓢ
Smalltalk/V286……………………[P]¥69,800Ⓢ
(PC,AT,PS/2,2Mメモリ,プロテクト版)
Smalltalk Goodies＃1…………[M][P]¥16,800Ⓢ
Smalltalk Goodies＃2,3…………[P]¥16,800Ⓢ
Smalltalk EGA……………………[P]¥16,800Ⓢ
Smalltalk Communication Kit[M][P]¥16,800Ⓢ
Smalltalk MAC……………………¥49,800

FTL MODULA-2

日本語マニュアル

Z80,8086,68000をカバーする総合プログラム環境を含む高速コンパイラ

CP/M80 Z80用……………ⓙⓈ[8]¥16,800
MS-DOS 8086用…………ⒿⓈ[M][P]¥19,800

68Kクロス・MS-DOS版………Ⓢ[M]¥68,000※
※近日発売

モジュール化機能をもつ最新言語です。

Pascalの設計者ヴィルトは，Pascalにモジュール化，並列処理，低レベル処理等の多くの改良を加え，より実用的な強力なModula-2を発表しました。
CやPascalで問題となっているプログラムの保守性は，プログラムの部品化，モジュール間の文法チェック機能により，モジュールが独立で行なえる様になります。

仕様

●走行環境
MS-DOS……V2.0以降 メモリ384K以上
CP/M……V2.2以降 58K CP/M以上
●8086系メモリモデル
プログラム64K，データ64K，ヒープは1Mのスモールとラージの2つのコンパイラが入っています。
●ライブラリ（Modula-2標準ライブラリ）
コルーチン，IOTRANSFERを含むフルセット。ガウス消去法によるマトリクス演算を含むライブラリーソースコード付属，MS-DOS版では，8087ライブラリ付属，割り込みベクトルトラップライブアラリでポップアップアプリケーションの作成が可能。
●漢字エディタ
エディタ内からコンパイル可能。ワードスターライクなコマンド，コンパイルエラー位置をエディタ内で指示。ポップアップメニューによるメインコマンド，マルチウンドゥ，マルチファイルエデットとファイル間ブロック転送，ユーザマクロ定義，インストールプログラム付。標準ではANSI画面コントロール。ソースプログラムも付属。
●ユーティリティ
ライブラリアン，トレーサ，テキスト圧縮，クロスリファレンス，アーカイバ，プロファイラ，GREP，パターンサーチ，CAT，COMPDIR，COMPBIN，COMPARE等，すべてソースコード含む。

高速Cインタプリタデバッガ，開発期間が大幅に短縮

TURBO C版出荷

C-terp V3.02

コンパイラーに100%コンパチブルなインタプリタです。C-terpはC言語ユーザーの入門から本格的なプログラマーにとっての最高のプログラム開発環境です。すばらしい開発統合環境がCの世界で実現できました。パソコンの前で何度もコンパイルを待っている間のいらだちはもうありません。もちろん漢字対応。

- K/R対応＋ANSI
- 高速コンパイラに劣らない実行速度(RUN/Cの4～50倍)
- 強力なデバッガー，トレーサ
- ポインタのチェック
- 分割コンパイル可能，仮想メモリ
- RAM640K必要
- ユーザファンクションのオブジェクトを取り込み可能
- 8087サポート
- デバッグスクリーンとアプリケーションスクリーンのスワッピング

ANSI版……………………ⒿⓈ[M]¥39,800
PC98……………………ⒿⓈ[M]¥39,800
IBMPC……………………ⒿⓈ[P]¥39,800
T.C版……………………ⒿⓈ[M][P]¥29,800

対応コンパイラ(MS-C，Lattice C，TURBO C)をご指定下さい。

解説書：MS-DOSデバッグ手法(ナツメ社)…¥1,500
C言語のデバッグ手法(啓学出版)……¥3,500

ROM化クロスコンパイラファミリ

HI-TECH C

ROM化ができるプロフェッショナルなCコンパイラで，多くのシステムハウスで使われています。ターゲットはZ80から68Kまで，OSはCP/M80，MS-DOS，CP/M86，UNIX，ANSI準拠のフルラインアップです。

8086用

MS-DOS版……………………ⒿⓈ[M]¥47,500
MSLINK版とHitechオプティマイズLINK版
CP/M86版……………………ⒿⓈ[6]¥47,500
強力なオプティマイズと高速コード出力

Z80用

CP/M80……………………ⒿⓈ[8]¥42,500
8bit用コンパイラは，使用するマシンのCP/Mメモリサイズにより2種類のバージョンがあります。
- V3.05…56K CP/M以上（TPA51K以上）
- V3.09…62K CP/M以上（TPA57K以上）

3.09は3.05にANSIの文法拡張を増強しており，そのために必要メモリーが増強しています。バージョンアップ，御購入の際には，利用されるマシンのCP/Mメモリサイズを御確認下さい。
MSDOSクロス……………………ⒿⓈ[M]¥47,500

N

South Wind Vol.11
★当社取扱いのハード、ソフトそして世界の動きなどの最新情報を満載した情報誌です。是非ご請求下さい。

■御注文は、在庫確認のうえ現金書留か銀行振込で、電話番号を必ず明記して下さい。(送料手数料￥970/1オーダーです）■振込銀行●三井銀行鶴見支店：当座5028608●富士銀行横浜駅前支店：当座24771●第一勧業銀行横浜西口支店：当座0117103●三和銀行横浜駅前支店：当座5890

より身近になった代理店網
●北海道/㈱ビーデー0166-61-2253●東北/アクセス山形㈲0236-44-9863●関東/㈱ビレッジセンター03 239 1093,関東電子㈱03 257-6301/フリータイム㈱03-739-2911/㈱ソルコム03-943-3441/㈱若松通商03-253-8521/㈱亜土電子工業03-257-2650/㈱シンフォニー03-443 0291/㈱ソフトウエアジャパン03-862-2765/㈱五橋研究所045-721-1885/㈱エムケイ0462-23-6613/ハートランドソフト0492-96-0306●中部/㈱三洋堂書店052-251-8334/㈱ワープクリエイトシステムズ052-961-5431/㈱システムイソムラ052-671-2651/黒田経営会計事務所0532-63-2294/スターランド0546-45-0707/㈱イナノ0559-62-0864/㈱エルクラフト0559-71-2015●大阪/㈱エムエスシー06-631-2435/アサイコンピュータサービス06-654-1616/ユニコムウエア㈲06-845 5475/共立電子産業㈱06-644-4666/ショップはる0742-71-1055/アルファブライト078-231-6333●中国/マイクロ出版㈱082-231-7736/㈱フタバ図書082-294-0184●九州/㈱エフアイシー092-622-5203/サンシー㈲0992-61-8250

テックサポート 045-314-1201 PM1:00～PM5:00
電話の前に、もう一度マニュアルをお読み下さい。（月～金）

SOUTHERN PACIFIC
株式会社 サザンパシフィック
〒220 横浜市西区南幸2丁目16-20 三和横浜ビル3F
TEL 045(314)9514／FAX 045(314)9840
JR横浜駅西口徒歩5分、営業時間 9:00～18:00日祝定休

資料請求No.333

フジキンソフト株式会社

セミナー会場新設！

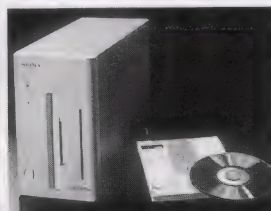
セミナー会場
●大阪センター　〒530 大阪市北区芝田2-6-30清和ビル
TEL.(06)376-4751

UNIX講習会

1. 入門コース
UNIXの一般的な使用に関する講習を行います。

2. 応用コース
UNIXのシステム運用に必要なファイル管理、ユーザ管理を中心とした講習を行います。

3. その他のコース
前述以外に、下記の様なアプリケーションの応用講習会をメーカー、ソフトハウス並びにシステムハウスとタイアップして開催致します。
■開発環境コース……………イメージシリーズ：イメージパートナー社他
■マイコン開発環境コース……C-ICE：コデジタル社(大阪コア社)他
■CAD環境コース……………Archi-CAD：建築システム社
CADIAN：テクノダイヤ社 他
■LAN環境コース

MO ディスクユニット ¥450,000　NWP-539

5インチのディスクに594MBもの情報を何回でも記録／消去でき、しかもフロッピーディスクと同様にリムーバルであるという特長を存分に駆使していただくことができます。又、別売の専用ボード(¥45,000)を使用することにより、※PC-9801で使用することができます。これらはすべて、**フジキンソフト**でお求め下さい。

※ディスクは通常、カートリッジから取り出せません。
※PC-9801は日本電気㈱の登録商標です。
■商品の価格には、消費税は含まれておりません。

※フジキンは納期でお応えします。

POPNEWS
NEWS
MO ディスクユニット

お求めは、創業60年を誇る
下記フジキングループへ！
(NEWS 代理店)

フジキンソフト主要業務

- ソニーのNEWS、popNEWS、及び周辺機器の販売
- 技術情報の売買 財日本テクノマート有資格会員
- 翻訳業務(技術翻訳も含む)
- ソフトウェアの製作、販売
- 教育用コンピュータコントロールシステムの製造販売
- 自動計測、制御システムの製作、販売
- バイオ用育苗システム「バイオトロン」の製造販売
- 流体制御機器の製造販売

コンピュータ・システム事業部

事業本部	TEL.06-376-4751	FAX.06-376-2342
EWSサポートグループ	TEL.06-782-2550	FAX.06-787-2257

札幌営業所開設　☎011-222-4681　FAX.011-231-3016

N

FUJIKIN SOFT

■本　社 営業総本部 事業本部 管理本部　〒530 大阪市北区芝田1-4-8(北阪急ビル)
☎大阪(06)372-7141(大代表)
FAX(06)375-0697 TELEX 523-4204

■東京本店営業本部　〒103 東京都中央区日本橋2-3-6(日土地ビル)
☎東京(03)273-0301(大代表)
FAX(03)278-0901 TELEX 222-6416

東日本ブロック

営業所	TEL	FAX
□八重洲営業所	☎東京(03)553-0381(代表)	FAX (03)552-9235
□芝中央営業所	☎東京(03)431-7841(代表)	FAX (03)431-8235
□新宿中央営業所	☎東京(03)366-8181(代表)	FAX (03)366-8189
□横浜中央営業所	☎横浜(045)662-4061(代表)	FAX(045)661-1090
□神奈川東営業所	☎神奈川(044)211-9351(代表)	FAX(044)211-9320
□京葉営業所	☎千葉(0472)46-7201(代表)	FAX(0472)46-6153
□日立営業所	☎日立(0294)24-2511(代表)	FAX(0294)24-2530
□仙台営業所	☎仙台(022)265-3081(代表)	FAX(0222)24-0176
□多摩出張所	☎国分寺(0423)25-6281(代表)	FAX(0423)25-6368
□湘南中央営業所	☎茅ヶ崎(0467)87-3811(代表)	FAX(0467)87-3888
□筑波出張所	☎茨城(0298)52-9021(代表)	FAX(0298)52-9023

中部日本ブロック

営業所	TEL	FAX
□名古屋営業所	☎名古屋(052)561-7681(代表)	FAX(052)561-7688

N

G-ISAMは、アプリケーションプログラムの開発効率を飛躍的に向上させるC言語用データ管理ライブラリーです。特筆すべきは、その高速性。将来のデータ量の増加にも処理スピードを低下させることなく、柔軟に対応できます。しかも、ファイルを意識しないで効率よくプログラム開発ができます。MS-DOSをはじめ、MS-WINDOWS、OS/2、UNIXなどの環境下で利用できるというきわめて高い汎用性を備え、さらにMS-WINDOWSとOS/2では、ダイナミックライブラリーとしても使用できます。

※UNIXオペレーティングシステムはAT&Tが開発し、ライセンスしています。
※MS-DOS、MS-WINDOWSはマイクロソフト社の商標です。
※OS/2はIBM社の商標です。

C言語用ISAMファイルライブラリー

G-ISAM

詳しい資料を差し上げます。ご希望の方は、下記宛にお電話または住所・会社名・部署名・商品名をハガキにご記入の上ご請求ください。

株式会社リコー　ソフトウェア事業部　〒112 東京都文京区小石川1-1-17とみん日生春日町ビル ☎03(815)7261 リコー

N

Microsoft®

マイクロソフト株式会社
〒160 東京都新宿区西新宿7-5-25 Kビルディング
電話(03)363-1201

Microsoft® CodeView®

MS-DOS, MS OS/2環境上のプログラムをデバッグする ウィンドウ指向ソースコードレベルデバッガCodeView

Microsoft CodeViewは、MS-DOSもしくはMS OS/2上のプログラム内の論理的な間違いを見つけ出すための強力なウィンドウ指向ソースコードレベルデバッガです。CodeView上で、作成されたプログラムを実際に動作させ、ソースコードの表示やアセンブリコードの表示、変数内容の表示、トレースや実行画面の表示などの機能を利用することでデバッグ効率が飛躍的に向上します。C、MASM、FORTRAN、BASICなど多数の言語に対応した操作性抜群のソースコードレベルデバッガは、マイクロソフトの言語シリーズと共に快適な開発環境を実現します。

MS OS/2対応

MS-DOSだけでなく、MS OS/2にも対応しました。MS OS/2の環境下では、理論上128Mバイトの実行ファイルをデバッグできます。また、マルチスレッドや、ダイナミックリンクライブラリのデバッグも可能となり、大規模な実用レベルのプログラムをデバッグすることが可能となっています。

抜群の操作性

CodeViewは、ドロップダウンメニューで処理を選択するウィンドウモードと、従来のSYMDEB方式のシーケンシャルモードをサポート、ユーザーの好みに合わせた操作を選択できます。ウィンドウモードでは、マウスによってコマンドメニューを選択、アプリケーション感覚で操作できます。また、リダイレクションコマンドで、ファイルや別のターミナルなどのデバイスからの操作(コマンドの入力と結果の出力)が可能となっています。また、CodeViewではデバッグ画面と実行画面を独立に管理、プログラムの実行でデバッグ画面が壊れることもありません。もちろん、MS-C5.1やQuickC1.1に添付されているグラフィックライブラリを利用したプログラムでも問題なくデバッグできます。CodeViewは抜群の操作性を提供します。

ソースコードレベルでデバッグ可能

CodeViewは、マイクロソフトの最新の言語シリーズ(MS-C5.1、MASM5.1、MS-FORTRAN4.1、QuickC1.1、Quick BASIC4.2)に対応、これらの言語で作成された実行ファイルは、ソースコードレベルでデバッグが可能です。また、複数言語(C+BASIC+FORTRANなど)によるプログラムなどもデバッグ可能、CのモジュールはCのソースコードで、BASICのモジュールでは、BASICのソースコードが表示されます。ソースプログラムそのものを見ながら、デバッグできるので、スムーズに問題点の発見をおこなえます。もちろん、ソースコードだけでなくアセンブラのニーモニックや、ソース+ニーモニックでの表示も可能です。

トレース、ブレークポイント、ウォッチ機能

デバッグはトレースが基本、トレースしながら理論上の誤りを見つけます。CodeViewでは、1ステップトレース、ブレークポイントの設定や、指定された条件が満たされると停止するウォッチポイントなど、さまざまな機能がサポートされています。
また、変数やメモリの内容、レジスタやフラグの内容を常に画面上に表示できるウォッチ機能もサポートしています。

80386のサポート

80386プロセッサ用に記述されたコードのデバッグをサポートしています。386レジスタの表示や、386命令のアセンブルやデコードまで可能になっています。

数値演算コプロセッサのサポート

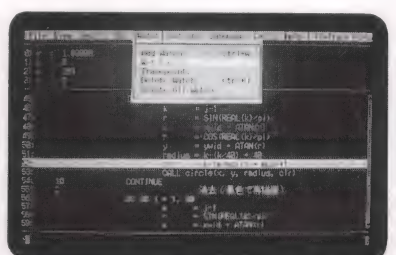
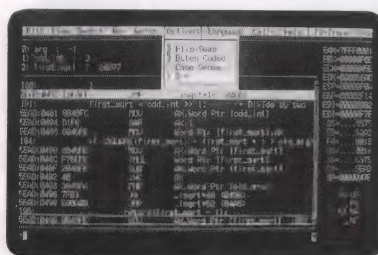
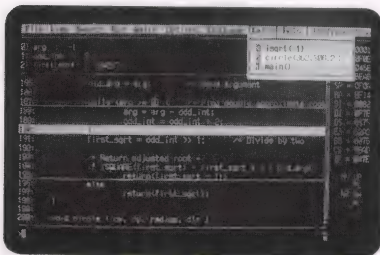
数値演算コプロセッサが装着されていない機種でも、それが装着されているかのように、8087レジスタ内容やコントロールワード、ステータスワードの表示が可能となりました。

オーバーレイとライブラリモジュールのデバッグ

オーバーレイを使うプログラムもデバッグ可能になり、ライブラリのデバッグもサポートされています。

CodeViewチュートリアルの添付

ユーザーの方に、CodeViewを使ったプログラムのデバッグを短期間で習得して頂けるようチュートリアルを用意しています。

CodeViewは以下のマイクロソフトの言語製品に標準添付されています。CodeViewは機種に依存したデバッガです。機種対応をご確認のうえ、御購入下さい。

Microsoft® C
Optimizing Compiler
Version 5.1

NEC・PC-9800/AXシリーズ対応　価格98,000円

Version 5.1

NEC・PC-9800/AXシリーズ対応　価格40,000円

Microsoft® FORTRAN
Optimizing Compiler
Version 4.1

NEC・PC-9800シリーズ対応　価格80,000円

●本広告に記載されている全商品の価格には、消費税は含まれておりません。

資料請求No.339

N

FUSION

ネットワークは進化する。

TCP/IP & XNS イーサネット

NFS

RFS

OEM UNIX,DOS,OS/2
ソース・ライセンス
移植・開発サービス
コンサルテーション

NEC PC9801
IBM PC/AT
& コンパチブル
バイナリー販売

VAX/VMS
4.×、5.0
アップデート可
バイナリー販売

日本無線 IPS200 iRMX286

富士通 FM-R MS-DOS

松下 CV-M MS-DOS

東芝 G-8000 OS/V J-3100 MS-DOS

日立 2050 E7700 E7300 UNIX

NTT DIPS V30 UNIX

フュージョンは
ひとつの宇宙に結合し、拡大し OSI …未来へ移行する。

異機種、異OS間の統合にシンプルでポータブルな解決、FNS(Fusion Network Software)がお応えします。

FNS基本構成

	TCP/IP	XNS
ユーザー・ユーティリティ	FTP, TELNET, SMTP, SERVER	SEND, RECV, RLOG, RX, SERVER, ECHOS, SETDATE
カーネル・プロトコル	TCP, UDP, ICMP, IP, ARP, GGP, クラスA/B/Cサブネット	XDG, XEP, XSP, XRP, Errorプロトコル

標準提供

NM(ネットワーク管理/テスト・ユーティリティ)

プロトコル解析、パケット/エラー リトライ/ジャムのカウントによるハードウェア障害解析、パフォーマンス・テスト

PDS(アプリケーション用ライブラリ)

4.2BSD通信ライブラリをモデルとした50以上のファンクション(FNS-PCではオプション)

ソースライセンス販売

FUSIONを自社マシンに移植されるユーザーのためにソースライセンス契約制度があります。移植サービスは当社技術陣により提供可能です。

FNS-VAX/VMS

FNS-VAX/VMSは全てのVAX/VMS システムをサポートいたします。

★豊富なオプション(＊はTCP/IP)

- TCP/IP、XNS両プロトコルの同時サポート
- DECNETとLANコントロール・ボードの同時的共用(DECボードのみ)
- UNIX 4.2BSDコンパチブルなRコマンド(RCP、RLOGIN、RSH、RUP TIME、RWHO)＊
- Mail/SMTP＊
- NFSサーバー＊
- DMR/DMV-11(DDCMP)ドライバー・サポート＊
- DDN/X.25ドライバー・サポート＊

★パッケージ価格

基本機能＋インストール＋年間サポート契約

環境
OS:VMS 4.x、5.0
プロセッサ:VAX/VMS全シリーズに対応
コントローラ:DEQNA、DEUNA、DELUA、DEBNT(以上DECNETとの同時共用可)、3COM、Interlan
バス:Qbus、Unibus、BIbus

パッケージ価格:¥700,000〜¥2,550,000

※ 定価に別途消費税3%が加算されます。

FNS-PC

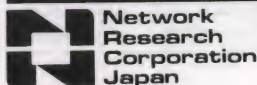
FNS-PCは、PC98、PC/ATとJ-3100などその互換機のMS-DOSパソコン用に、以下の充実した異機種間接続機能を実現します。

プロトコル:TCP/IP　XNS
仮想端末:日本語telnet、rlogin　rlog
(エミュレーション……VT100)
ファイル転送:ftp、ftpサーバ、rcp　send/recv
遠隔実行:rsh　rx
ネットワークマネジメント:パケットモニター、パフォーマンステスト、障害検出
ファイルシェア:NFSクライアント(オプション)
ライブラリ:4.2BSD相当　ソケットライブラリ(オプション)

基本構成価格
PC98シリーズ:¥180,000(コントロールボード込み)
PC/AT、AX、コンパチ機:¥120,000(コントロールボード別売)

オプション価格
NFSクライアント:¥ 80,000
ソケットライブラリ:¥200,000

ローカルエリアネットワークのトータルソリューションを提供する

Network Research Corporation Japan

ネットワークリサーチコーポレーションジャパン株式会社

〒160 東京都新宿区新宿4-1-9 新宿ユースビル6階 TEL.03-356-3561 FAX.03-356-2190

販売代理店

- 日本アイ・ディ・アイ株式会社 TEL.044-922-0711
- アイ・テイ・エス ジャパン株式会社 TEL.03-343-1811
- 情報技術開発株式会社 TEL.06-535-1701
- データコントロルズ株式会社 TEL.06-443-3260
- 菱洋エレクトロ株式会社 TEL.03-546-5100

N

N

新発売!

OS9/68000版の新バージョンがFA、LAプログラマーの世界を変える。

リレーショナル・データベース

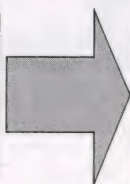
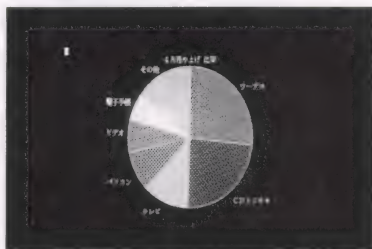
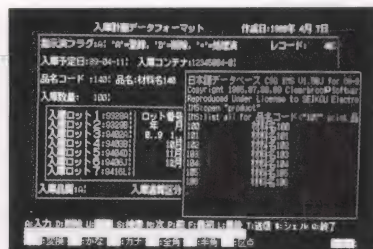
CSG IMS™ Ver. 2.0

シーエスジー・アイエムエス

データベースCSG IMSとCがドッキングすることで開発効率が驚くほど向上しました。

あなたは50%以上の開発効率向上に背を向けますか?

視点を変えると新しい世界が見えてくる。

あなたは、
- ●FA、LAの応用プログラムをC言語だけで開発していませんか。
- ●Cの世界だけで、よその世界が見えなくなっていないでしょうか。
- ●データベース言語は、OAのツールだという先入観を持っていませんか。
- ●データベース言語に、リアルタイム、マルチタスク処理ができないと思っていませんか。
- ●何も疑問を持たずに、大量のデータ処理をCだけで、大量の画面操作をCだけで開発していないでしょうか。
- ●他人が読めない保守性の悪いプログラムになっていないでしょうか。

〔新機能〕
- ■ランゲージインターフェース機能追加。IMSよりC、アセンブラのプログラムをコール可能。
- ■グラフィックライブラリーを追加。星光電子仕様、フォークス仕様の各基本関数をサポート。
- ■データベース定義操作が簡素化され操作性向上。
- ■IMSスタート時のメニュー画面の操作性向上。カーソルとリターンキーで機能選択。メニュー画面用のプログラムソースコードが添付されます。
- ■フルスクリーン・エディタTXの機能強化。

●グラフィックス可能 IMSよりCで書かれたグラフィックス・サブルーチンをコールして円グラフを描画。

●WINDOWS16、WINDOWS98と組合せて、例えばIMS応用プログラムを複数本同時実行、同時表示。

〔お知らせ〕新バージョン発売を機会に商品形態を次の様に変更いたします。

適応機種	価格	販売
FM11、FM16β、PC9801シリーズ	¥118,000	好評発売中
FM-R、Panacom Mシリーズ	¥118,000	新発売

- ●FM11、FM16β、PC-9801、FM-R、Panacom Mシリーズには、別売りの㈱フォークス製のOS-9/68000、OS-9/68020が必要です。
- ●VME対応版は各ベンダーとOEM交渉中。
- ●68000版の登録ユーザー様にはバージョンアップのDMを発行しております。ユーザー登録されてない方は、至急登録してください。
- ●6809版は今回のバージョンアップの対象となりません。

デモを実施中

東京地区(秋葉原) ㈱星光電子内
……………電話予約が必要です。
指定パソコンショップ等でのデモ/セミナーを準備中です。

代理店、VAR業者募集中

データベース+通信システムのシステム開発、システム販売ができる技術力のある協力会社を求めています。詳細は、担当:浅井にご相談ください。

IMSはカナダCSG(Clearbrook Software Group)社が開発した、言語型の本格的なリレーショナル・データベースです。その仕様は柔軟性に富み、応用のきく、気のきいたデータベースです。言語そのものはBASICに近い仕様になっているため、だれでも簡単に習得できます。データ構造には、検索を重視してB+Tree方式を採用、驚くほどの高速検索とデータの安全性を誇ります。

言語型のデータベースというと難しいという印象がありますが、初心者でも簡単に操作できるように、メニュー画面方式になっています。さらに、本製品には簡単に操作できる2種類のフォーマッタが付属しています。フォーマッタとは、データベースのプログラムを自動的に作成するプログラムです。住所録や簡単な顧客管理程度であれば、画面のレイアウトや印刷するフォーマットを定義するだけで自動的にソースプログラムを生成してくれます。もちろん、このソースプログラムは、付属のスクリーンエディタで自由に変更できます。初心者から高度な利用を考えている上級者にいたるまで、幅広いニーズに対応します。

また、OS-9のマルチユーザ、マルチタスク機能により、TSSやLANなどを用いて複数のコンピュータで1つのデータベースを共用できます。さらに電話回線を使った遠隔操作も可能です。「データの入力をみんなでしたい」、「出張先からデータを取り出したい」こんな願いに応えます。

OS-9/68000

- ●OS9-MRX01 (FMR&PanacomM用1MBのRAM付き)·¥320,000
- ●OS9-MRX04 (FMR&PanacomM用4MBのRAM付き)·¥480,000
- ●OS9-68K98X (PC98VX/UX/RX)……¥198,000
- ●OS9-68K98 (PC98/E/F/M/VM用)……¥198,000
- ●OS9-98NET2⁺ (OS9/68000CPU+OS9/NETカード)·¥288,000
- ●OS9-68KBAS (BASIC)……¥ 60,000
- ●OS9-CONV02 (FM16β、PC98用OS-9とDOSのファイル変換、OS-9で動作)……¥ 19,800
- ●DOS-CONV01 (FM16β用OS9とDOSのファイル変換、DOSで動作)……¥ 19,800
- ●DOS-CONV02 (PC98用OS9とDOSのファイル変換、DOSで動作)……¥ 19,800
- ●OS9-CDISK2 (キャッシュディスク・ドライバ)……¥ 16,000
- ●OS9-WINDOW98 (98版OSKのマルチウィンドウ・ドライバ)……¥ 39,800
- ●OS9-WINDOW16 (16β版OSKのマルチウィンドウ・ドライバ)·¥ 39,800
- ●OS9-SHL (OS9/680X0用拡張SHELL)……¥ 12,800
- ●OS9-DBG (OS9/680X0のドライバ用デバッガ)……¥ 70,000
- ●OS9-CDBG (V2.2用Cソースコードレベルデバッガ)……¥ 80,000
- ●Q-Tools (MSDOS用ターミナルソフト等)……¥ 28,000
- ●OS9-VSED (フルスクリーンエディタ)……¥ 28,000
- ●OS9-SPL (プリントスプーラー)……¥ 40,000

※OS-9/6809関連のソフト/ハードについては、別途カタログをご請求してください。

上記商品の価格は全て消費税別の表示です。

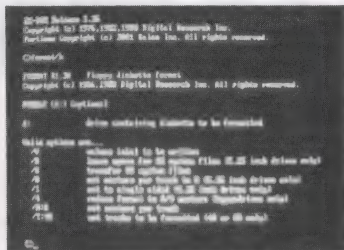
AUTHORIZED OS-9 SOFTWARE IMPLEMENTOR
株式会社 星光電子
〒101 東京都千代田区外神田6—5—11 長谷川ビル
TEL. 03(832)6000代 FAX. 03(833)6100

主な取扱店 ●T・ZONE(秋葉原)☎03-257-2650〈担当〉佐藤 ●J&Pコスモランド(大阪)☎06-634-3111〈担当〉中川 ●ソフトクリエイト(渋谷)☎03-486-6541〈担当〉津田

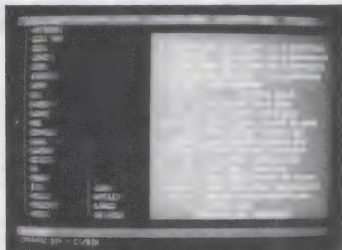
N

結論 OSは簡単であるべきです。

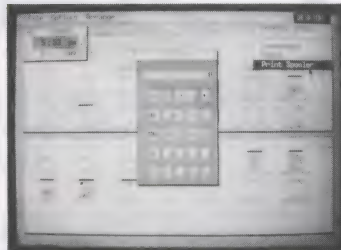
16ビットディスクオペレーティングシステムのご紹介

まだ8ビットOSですか？　EZ-DOSはインテル社製CPU搭載機用に開発された16ビットOS。
使いなれたアプリケーションソフト、そして世界の資産ともいえるXT/ATのアプリケーションソフトの数々を有効にご利用頂ける強力でしかも使いがってのやさしい民主的OSです。

EZ-DOS

EZ-DOSの本体で、OSのROM化を基本コンセプトに16ビットで開発されています。従来のDOSソフトやハードのサポートに加え、未知の可能性を秘めたOSです。

- DOS1.0~4.0のアプリケーションをフルカバー。（下記コンパチリスト参照）
- 512MBまでパティション無しでハードディスクをサポート。データーベース構築など今後必需になる大型ハードディスクに威力を発揮します。
- パスワードファイルプロテクト。見せられないファイルや、どうしても触れられたくないファイルを対象に、任意のパスワードをつかってプロテクトを設定できます。
- DOSコマンドのオンラインヘルプ機能。コマンドやオプションを解説します。マニュアルを開く手間がかかりません。
- LIM4.0を標準サポート。

Top DOS

EZ-DOSの外部コマンドの一つで、強力な統合ユーティリティーです。ディスク容量の1,000MBまでファイルとディレクトリーをコントロールできます。ハードディスクのファイル整理に最適です。

- TREEコマンド。サブディレクトリー形態をグラフィックで表示し、矢印キーにてディレクトリーを移動、カレントディレクトリー内のファイルを表示します。ドライブ変更、ファイルのコピー、削除、ソートなどDOSコマンドを簡単なキータッチで実現します。
- Editorコマンド。パワフルなフルスクリーンエディターを標準装備していますので、CONFIG.SYSやAUTOEXEC.BATの作成はもちろん、プログラミングやTEXTファイルの作成などミニワープロとして利用できます。オンラインヘルプ機能付き。(54,000bytesまで)
- サーチ及びマクロ機能をサポート。

GEM/3 Desktop

グラフィック、ウィンドウをオペレートする環境ユーティリティーです。

- ドライブの指定後、上下に個々のドライブの内容をアイコン画面で表示します。同一ドライブの各サブディレクトリーも指定できますので、マウスを使ってドライブ間やドライブ内でDOSコマンドやファイルの取扱いがより簡素化できます。
- GEMシリーズのアプリケーション及びDOSアプリケーションを、ワンタッチで起動復帰させるユーザーインターフェイスです。
- アプリケーション、データー、プログラム等のファイルを、内容に沿ったアイコンで設定できます。テキスト画面に変更も容易です。サーチ、ズーム、スクロール機能により、すみずみまでファイルの管理ができます。
- 電卓、アラーム付きデジタル時計、プリントスプローラーなどのデスクアクセサリーを装備しています。

True BASIC

BASICの開発者、ジョンケメリーとトムカーツによって書かれた現代版BASIC。

- ANSIスタンダードに準拠する、ストラクチャードBASIC。PCで書かれたプログラムが、MacやAmigaなどの他のマシーンで走ります。
- ライブラリーと共にコンパイル可能。◆マルチウィンドーをサポート。
- SELECT CASE,IF-THEN-ELSE IF,及びDO-LOOPなどのコントロールストラクチャーをサポート。◆マトリックスI/Oなどのマトリックスマスを装備。
- オンラインヘルプ、及びオンラインシンタックスチェック、インタープリターとして容易にプログラミングできます。
- デベロッパーズツールキットとして8シリーズを用意しました。（各ライブラリー別売）
- Runtimepackage.各マシーン用にコンパイルするアドオンソフト。(機能限定版、別売)
- Runtimepro.全ライブラリーをサポートするRuntimepackageのプロユース。(別売)

●コンパチリスト（弊社調べ）

Autocad Release 9
Boeing Graph 3D 4.0
Carbon Copy Plus 4.0
ChartMaster 6.21
Clipper Summer 87
COPY2PC 4.01
COPY2PC 5.0
CrossTalk Mk4 1.01
CrossTalk XVI 3.61
Dac Easy Accounting 2.0
dBase III Plus 1.1
Desqview 2.01
Display Write 4 1.0
EGA Paint 2005
Excel 2.0
Fastback Plus 1.01
Flight Simulator 2.13
Flight Simulator 3.0 Beta
Freelance Plus 2.0
GEM Desktop 3.0
Generic CADD Level 3 1.0
HALO DPE 1.20.14
Harvard Graphics 2.10
Lightning 4.8
Lotus 1-2-3 2.01
Lucid 3-D 1.0
Mace Utilities 4.10c
Multimate Advantage II 1.0
Norton Utilities Adv. Ed. 4.0
PageMaker 1.04
Paradox 2.0
PC Tools 4.22
PFINISH 1.10
PFS：1st Choice 2.01
PFS：Professional Write 2.0
Prokey 4.0
Publisher's Paintbrush 1.53
Q&A 3.0
Quattro 1.0
Rapidfile 1.2
Rbase for DOS 2.1
Ready! 1.00e
Sidekick 1.56a
Smartcom III 1.0
Superkey 1.16a
Symphony 2.0
Turbo Pascal 4.0
Ventura Publisher 1.1
VersaCad 2D 5.3
Windows 2.03
Word 4.0
WordPerfect 4.2
Wordstar 2000 Plus 3.00

※上記コンパチリストに記載されているソフト名は各社の登録商標です。

2000's EZ-DOS™

オペレーティングシステム

OEM供給

弊社は既にラップトップ及びデスクトップメーカーへの販売体制を確立し、技術資料等の供給を開始しました。
又、ソフトメーカー及びハードディスクメーカーへのOEM供給も近日中に開始いたします。

パッケージ販売

EZ-DOSパッケージは、御近くのパソコンショップで御買い求め頂くか、弊社まで御問合せ下さい。

VANET バネット株式会社
〒141 東京都品川区東五反田1丁目25−19
東建島津山南ハイツ102号
☎03−280−2645 担当／渡部
FAX03 440−0136

N

使ってわかるこの便利環境

MS-DOSのバッチ機能と同様な、C-terpの一連のコマンドをファイルから自動的に実行するコマンドです。

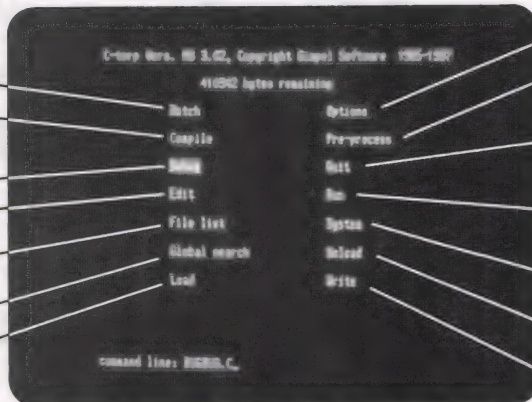
指定したファイルをコンパイル（中間コードに落とす）します。RUNコマンドで実行する前に、シンタックスエラーのチェックに利用します。

デバッグ・モードで実行。

エディタを起動します。

現在ロードされているファイル全部のリストを表示します。

文字列のサーチを現在ロードされている全ファイルで、行番号とともに表示します。

ファイルをメモリ上にロードします。

サブメニューで種々の設定を変えられます。

プリ・プロセッサを呼び出します。これで、どの様にプログラムが変換されていくかを見ることができます。

C-terpを終了し、システムに戻ります。Writeしていないファイルは Writeするか1つ1つ聞いてきます。

プログラムの実行。エラーが起きると1キーでエディタに入りエラー箇所にカーソルが行きます。

DOSのレベルに戻って、DOSコマンドを使用できます。EXITでまたC-terpに復帰します。

ロードされているファイルをメインメモリから消去し、メインメモリを解放します。

指定されたファイルをディスクに書きます。

概要

C-terpは、C言語のインタプリタです。それ自身をお使いのCコンパイラで再コンパイルして、Cコンパイラのラージモデル・ライブラリとリンクして使用します。実際にCコンパイラのライブラリを中間コードに落として実行していますのでCコンパイラとのコンパチビリティが実現できます。

C-terpは、C言語の入門者からプログラマまで使えるすばらしい開発環境です。C-terpは、実行時にまずソースを中間コードに落し、それから高速な中間コードインタープリタが実行されます。時間のかかるリンクなしで即実行しますのでパソコンの前でコンパイルを待っている間のいらだちはもうありません。もちろん漢字には対応していますし、マニュアルも日本語です。

C-terpは、コンパイラと併用できる様に設計されています。C-terpでデバッグしながら作ったプログラムを最終的にコンパイラにかけたり、コンパイラで作ったCの関数をC-terpのなかに外部関数として取り込んで、C-terpの中からそれを利用したりできます。

デバッグ機能

C-terpでは、対話型とトレースの２つの方法によるデバッグ法があります。

●対話型

デバッグ・モードに入るのには次の６つの方法があります。

1. breakpt()関数を実行する。
2. 恒時的ブレークポイントに出会う。
3. 一時的ブレークポイントに出会う。
4. ブレークキーを押す。
5. 実行時エラーが発生する。
6. メインメニューでデバッグ・モードを選択する。

各コマンドで、実行を再開したりプログラムの出力スクリーンに切り換えたり、操作がわからなくなった時にはデバッグ・ヘルプを表示できます。カーソルの位置まで実行したり現在の関数を実行し抜け出したりもできます。デバッグ時のウインドウ分割のサイズの変更もできます。１ステップ実行、関数コールのスキップ、制御段階の関数をトレース表示、恒時的なブレークポイントのセットまたは解除、式の値を表示、デバッグを中断してEDITに入る、ブレークポイントまでの実行、Watchコマンドなどがサポートされてデバッグ機能が充実しています。

●トレース

BASICの'TRON'の様に簡単にトレースをするようなスイッチがC-terpでは用意されています。トレースの出力は、通常スクリーンですがオプションメニューのトレースファイルオプションでファイルへ出力もできます。

トレースは、プログラム内に関数trace()を書いて行ないます。trace()関数の引数により、いくつかのトレース・モードを選べます。

教育用として

C-terpの開発者である、Dr.James F GimpelはLehigh大学の彼の授業で、C-terpを用いています。教育用ディスカウントライセンスも行なっていますので、お問い合わせください。

Turbo C版

Turbo Cのグラフィックライブラリを使ったプログラムのデバッグも可能です。

N

ANSI版		PC9801専用版		IBM-PC専用版	
Microsoft C用	ⒿⓈⓂ¥39,800	Microsoft C用	ⒿⓈⓂ¥39,800	Microsoft C用	ⒿⓈⓅ¥39,800
Lattice C用	ⒿⓈⓂ¥39,800	Lattice C用	ⒿⓈⓂ¥39,800	Lattice C用	ⒿⓈⓅ¥39,800
		Turbo C V1.5用	ⒿⓈⓂ¥29,800	Turbo C V1.5用	ⒿⓈⓅ¥29,800

Ⓙ和文マニュアル　Ⓢメディア変換料がかからない商品　ⓂMS-DOS版　ⓅPC-DOS版

South Wind Vol.11

当社取り扱いのハード、ソフト全商品、及び最新情報、記事を満載した情報誌「サウス・ウィンドVol.11」が出来上がりました。ご請求下さい。

SOUTHERN PACIFIC
株式会社サザンパシフィック
〒220 横浜市西区南幸2丁目16-20 三和横浜ビル3F
TEL 045(314)9514／FAX 045(314)9840
JR横浜駅西口徒歩5分、営業時間 9:00～17:00 日祝定休

資料請求No.346

新発売　33MHz DS386 AT互換機

DS386/33はAMI社が開発した新製品を使用した超高速の32Bitコンピュータです。その性能はミニコンのSun4/260、Apollo DN4500、VAX8650より高速です。メモリは16MBまで32Bitで拡張できます。

CPU80386 32BIT AT互換機(AMI)

- DS386-33(33MHz)
  メモリ4MB、64KBキャッシュ、1167、3167
- DS386-25(25MHz)
  メモリ1MB、64KBキャッシュ、1167、3167
- DS386-20(20MHz)
  メモリ1MB、64KBキャッシュ、1167、3167
- DS386-16(16MHz)

CPU80286 16BIT AT互換機(ATRONICS)

- DS286-10(10MHz) メモリ640KB
- DS286-12(12MHz) メモリ640KB

※AMI社、ATRONICS社は米国での一流のコンピュータメーカです。

Persoft

Smartem

SmartemはUSAで究極のターミナルエミュレーターとして広く使用されております。メインフレームシステムのソフトを他のコンピュータからアクセスしてアップロード、ダウンロードできます。異ったコンピュータと情報を交換します。

Smartemはコミュニケーションとエミュレーションの両方の機能を持っております。ReGISやTektronixのグラフィックを使用できますのでCAD/CAMをメインフレームとPCとの間でやりとりできます。その他DECのテキスト ターミナルとしてDECのフルスクリーンエディターを利用できます。電話を通してテキストとバイナリーファイルをトランスファーできます。

Smartmove

- 正確なDEC VT100、VT102、VT152のエミュレーター
- PFファンキーとPFI "Gdd" keyが使えます。
- スクロールレートは可変
- DECのアンサーバックが出ます。
- USとUK文字をサポート
- DEC LEDインディケータ
- DECの全プリント

Smarterm 240

- DEC VT340ReGISグラフィックをサポートしております。
  ダイナミックコンプレッションができます。
- Tektronixグラフィックをサポート
- スケーリングでスクリーン上に一度で全イメージを表示できます。
- ズーミングは1,024×780の解像度の部分表示が可能です。

Smarterm 400

D410、D400、D215、214、211、210、200、100をサポートしています。

Lathy F77L FORTRAN

F77LフォートランはUSAの各コンピュータ専門誌上で最高の評価を得ております。F77L-EM/16、F77L-EM/32は、それぞれ16Bit、32Bitのプロテクトモードをサポートしております。特にDEC VAXやIBM VSのメインフレームでのフォートランと共通に使えるように考慮されております。コンパイル速度と実行速度は、他社に比べて最高速といわれています。

- ANSI FORTRAN77に完全準拠
- "NAME LIST" ができます。
- メインフレームプログラムのダウンロードとアップロードが可能です。
- ソースオンデバッガ付
- MSC、Turbo C、Lattice C その他のインターフェース付
- 8087/80287/80387のエミュレート付なので数値演算素子なしで動かせます。(F77Lのみ)
- プログラムごとにスタックは、ローカル変数のために64Kまでとれます。
- 配列は7次元まで、配当のサイズは640K(F77L)、15MB(F77L-EM/16)、4GB(F77L-EM/32)です。

品名 F77L
F77L-EM/16
F77L-EM/32
LAHEY F77L PLINK OVERLAY LINKER
LAHEY/A.I. OPERATING SYSTEM

Asyst

Asystは高度の科学データ処理ソフトです。初心者にも容易に使用できるマクロでユーザは自由にプログラムできます。
LOTUS 1-2-3のファイルの読み書きもできます。

1. データアクジション機能

AsystはA/D、D/A変換ボードを用いたデータの取入れと送り出しを行います。DMAモードでは同時に最大DMA5チャンネルまで使え、またこのチャンネルは異なったコンピュータ上で利用できます。
高速のA/DボードからデスクまでDMAを使うことができます。もし高速のデータアクジションと同時にデータの表示を求める時は、DATAQ Waveform Scrollerボードを用いて下さい。

2. アナリシス機能

- カーブフィッティング　●配列演算　●統　計
- 波形演算　FFT、逆FFT、2次元FFT、スムージング、フィルターリング、ピークディテクタ、コンボリューション、微分、積分

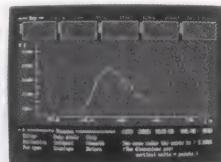
例) 1,024点実数FFTを0.47秒(16MHz 386cpu)で行います。

3. グラフィック機能

Asystはオートプロットの機能を持っています。各種の関連しているデータ、関連していないデータを自動的に対照表示できます。データが不連続でもカーブフィッティングができます。データにいろいろな処理をして幾つかのグラフをひとつのスクリーンに表示することができます。データはズーミング、等高線3Dのシュミレートが可能です。

Asystant

AsystantはAsystと殆んど同様の機能をもち、さらに完全なメニュー画面により操作ができます。これは簡単な応用に最適です。
ASYSTANT PLUSはD/A、A/Oボード及びデジタルI/Oをサポートしています。

品名 **ASYST** モジュール 1、2、3、4
組合せモジュール (1、2、3)、(1、2、4)、(1、2)、(3)、(4)、(3、4)
ASYSTANT PLUS
ASYSTANT

GSS Graphic

GSSはDOS、OS/2、XENIX、UNIX用のグラフィックソフトのメーカーであり、又、IBMにGDTをOEM供給しております。
DOS用のGDTはソースコードでOS/2と互換性があり、再リンクするだけで稼動します。又、OS/2のプレゼンテーションマネージャーのウインドウで稼動します。
Metafile InterpreterはANSIのバーチュアルデバイスメタファイルをサポートしています。
GSS、X/386はサン、アポロ、DEC、IBM、その他の80386 Unixマシン上のXウインドウにアクセスします。又、TCP/IPをサポートしています。又、TCP/IPをサポートしています。GKS Kernel Systemもあります。

品名 ●Graphics Development Toolkit
●GKS Kernel System

P

WYSE

新発売　WYSE 386/25MHz(5.75MIPS)

WYSEコンピュータは、米国のOAシステムとして最も高品質で、デザインも良く、高速であるとして広く用いられています。
コンピュータの高速化に伴い、386/25MHz、286/16MHzを販売しました。その性能は、5.75MIPSでIBM PS/2-70の5.74、Compaq386/25の5.50を凌いでいます。
価格は、安く設定されており、ターミナルのWY60は、販売分野で権威のあるVARBUSINESSで"Best of Sell"やComputer Dealerで"Product of the Year"を確保しています。

CPU80386 32BIT パソコン
- WY-3225-01(25MHz)
- WY-3225-150T 150MB HDD(25MHz)
- WY-3225-300T 300MB HDD(25MHz)
- WY-3216-01(16MHz)
- WY-3216-40 40MB HDD(16MHz)

CPU80286 16BIT パソコン
- WY-2116-01(16MHz)
- WY-2116-40 40MB HDD(16MHz)
- WY-2112-01(12.5MHz)
- WY-2112-40 40MB HDD(12.5MHz)
- WY-2200-01(10MHz)
- WY-2200-40 40MB HDD(20MB有)(10MHz)

NEC PC-98シリーズ用 ATソフトウェア(漢字対応)

コンバージョンソフト付 漢字は/Jでコンパイル
そのままでAT用ソフトとして御利用出来ます。

ソフト	機種	価格
MICROSOFT C	M.98	¥60,000
MICROSOFT QUICK C	M.98	¥24,000
MICROSOFT MASM	M.98	¥34,000
MICROSOFT PASCAL	M.98	¥60,000
MICROSOFT WORD	M.98	¥67,000
TURBO C	M.98	¥30,000
TURBO PASCAL	M.98	¥30,000
TURBO PROLOG	M.98	¥30,000
F77 LAHEY FORTRAN	M.98	¥110,000
REDUCE	M.98	¥108,000

(全て最新版です。)

CPU80386 33MHz 大容量650MB 新発売

世界的に有名な32bit ATコンパチ機

32bit機の世界の基準マシーンです。コンパックはIBMと並び称せられるパソコンです。その性能には定評があります。USAではソフトウェアやハードウェアのメーカーはコンパック機での動作確認が必須条件となっております。
コンパックはインテル82385の32KB SRAM(35ms)付キャッシュメモリコントローラーを使用している為に、メモリアクセスは非常に高速となっています。IBMはDRAMをページメモリで使用しています。又、1:1のインターリーブであり、又、ディスクキャッシュが付いています。

Compaq 386/20 Model110とIBM PS/2 Model70-121の比較では(　　)内は386/25との比較

1. CPU/メモリテストでは、34.4%(71%)コンパックが高速です。
2. ハードディスクテストでは、47%(71.4%)コンパックが高速です。
3. ハードディスクテストでは、ディスクキャッシュ付の場合は2.5倍(2.8倍)コンパックが高速です。
4. 387付の場合、Auto Cad上でのテストでは27.3%(47.4%)コンパックが高速です。
5. コンパックのVGAは、IBM VGAに比べて50%性能がUPしています。特にBLOSの実行スピードテキストのスクロールや、グラフィックスピードがすぐれています。

新発売

- COMPAQ Portable Ⅲ MODEL20 (12MHz)
- COMPAQ Deskpro 286 MODEL1 (12MHz)
- COMPAQ Portable 386 MODEL40 (16MHz)
- COMPAQ Deskpro 386S MODEL20 (16MHz)
- COMPAQ Deskpro 386 MODEL40 (16MHz)
- COMPAQ Deskpro 386/20 MODEL60 (20MHz)
- COMPAQ Deskpro 386/25 MODEL110 (25MHz)
- COMPAQ Deskpro 386/20e MODEL40 (20MHz)
- COMPAQ LAPTOP SLT/286 MODEL40 (12MHz)
- COMPAQ Deskpro 386/33 MODEL84 (33MHz)
- COMPAQ Deskpro 386/33 MODEL320 (33MHz)
- COMPAQ Deskpro 386/33 MODEL650 (33MHz)

COMPAQ™

新発売 Release2.3

SCO XENIX SYSTEM V

この新しいRelease2.3は、AT & TのUnix System V/386 Release3.2の機能をサポートしております。
UnixのSVVSをパスしていますので、SVID Release 3に100%準拠しています。Release2.3からUnixとソース及びバイナリーの互換性が保証されました。
イーサネットの為のSCO TCP/IPのRuntime及び開発システムが付いています。又、X/Open CAEがサポートされています。
SCOのコンパイラーは非常に好評で、他のCコンパイラーよりもメモリーが半分で済みます。
SCOはUnixの他のヴァージョンより実行速度が25%～100%高速と言われています。
ESDI、SCSIがサポートされています。

- Complete SCO XENIX SystemV
- Operating System
- Development System
- Text Processing System

SCO総代理店

MultiView

デスクトップパブリッシング用マルチスキャンモニター(低価格)超高解像度ポートレイト(800×1,000)とランドスケープモード(1,024×768)のハードスイッチ付です。
IBM XT、AT、PS/2互換機に接続できます。

超高速PS-388：
ポストスクリプトコントローラカード(32bit、10MIPS)
Grafix Pro：
ディスプレイコントローラボード(800×1,000、1,024×768 CGA、MDA、EGA、VGA)

Ventura、Lotus、MS Window、Auto CAD等に最適です。

PRINCETON PUBLISHING LABS 総代理店

P

IBM•AX系パソコンソフトウェア SCO. Media cybernetics. Graphic Software. ALR総代理店

MICROSOFT™

●MICROSOFT
Quick Basic Basica コンパチ、構造化……P¥19,000
MS PASCAL MSCとリンク可……M¥45,000
Microsoft Windows Development kit Window 開発キット……P¥87,000
Microsoft muMath83 代数学、MuSimp を含む……P¥55,000
Microsoft Operating System/2 Programmer's Toolkit……P¥61,000
MICROSOFT BASIC コンパイラー……P¥54,000
MICROSOFT EXCEL……P¥75,000
Microsoft Window 286……P¥19,000
Microsoft Windows 386……P¥33,000
Microsoft Basic Interperter……P¥58,000
Microsoft Project……P¥42,000

●Cコンパイラー
Microsoft C 新製品、世界最速(V.4.0より30%高速)……M¥70,000
QUICK C付(統合エディター、コンパイラー、ソースレベル、デバッガー、80287サポート)
Lattice C……M¥60,000
Microsoft Quick C 最新版……P¥19,000

●インタープリンター
Interactive C (Impaco Assoc) Computer Langauge 話推薦……P¥56,000
Instant C (Rational Sys) 最高速インタープリター……M¥82,000
(C-terpの数倍高速)、リアルタイムソースレベルデバッガ、エディタ付

●Cユーティリティ
C++ (Guidelines Software)……M¥42,000
C To オブジェクト指向 C Translator
Greenleaf Functions (Greenleaf)……M¥39,000
有名なCライブラリー集、高速、ソース付
Greenleaf Superfunction LIM4.00の活用ソフト……P¥60,000
Greenleaf コミュニケーション ライブラリー、ソース付……P¥48,000
Greenleaf Data Windows オーバレイウィンドー、254ロジカルウィンドー、ソース付、OS/2用あり……P¥64,000
PC Lint (Gimpel) 使いやすい Lint……M¥27,000
C Scientific Subroutines (Peerless)……M¥41,000
Vermont VIEWS 水平、垂直スクロール、nest、ウィンドーより大きいデータ入力 OS/2、Unix、XEMX、VMS にポート可……P¥62,000(ソース付 ¥120,000)

●フォートラン
F77L Fortran (Lahey) 最新版……M¥105,000
コンパイル速度はMSの4倍以上
Personal Fortran 77 (Lahey) ANSI 100%、超高速……M¥35,000
F77L-EM/32 32bit プロテクトモード用……P¥170,000
NDP Fortran (Micro way) 32bit……P¥125,000
Scientific Subroutine Libtrary (Wiley) C or Fortan……P¥50,000
Microsoft FORTRAN……M¥72,000
MS Cと混用可、ANSIフォーラン100%コンパチ、世界最高速、
R/M FORTRAN (Ryan Mcfarland) 高速……M¥104,000
F77L-EM/16 プロテクタモード用……M¥140,000

●デバッガー、アセンブラー
Advanse Trace-86 (Morgen Computing) デバッガ、プロ用……P¥43,000
Intrpreter Assembler付
Microsoft MACRO Assembler 最新版……M¥29,000
MASM、SYMDEB、LINK 高速化
MS Quick BASIC、C、FORTRAN、その他とのリンク用プログラム付

●エディター、リンカー
Brief (Solution System) 高速、最良のエディター……P¥42,000
(ウィンドー、多数回アンドウ、マルチプルファイル、コマンモード)
EPSILON (Lugaru) EMACS like、高速……P¥45,000
P Link86 PLUS (Pheonx) オーバーレイリンカ……M¥67,000
EMACS (Unipress)……P¥78,000
強力、マルチファイル、ウィンドー、他にソースも有り、最良
VEDIT PLUS (Compu View Products)……OS/2 P¥42,000
マルチプルファイル、ウィンドー分割

●ワープロ
Lotus Manuscript……P¥84,000
Wordstar 2000⁺ (Micropro)……P¥62,000
Wordstar professional 最新版……P¥62,000
Word (Microsoft) 習得容易、高機能……P¥52,000
XY WRITE III (Xy Quest) 高速度高機能、最新版……P¥50,000
Word Perfect (Satellite) 高機能、office用に最適……P¥62,000

●通信ソフト・ネットワーク
Crosstalk XVI (Microsoft) 標準的通信ソフト、高信頼性……M¥29,000
Smarterm (Persoft) VT100、VT240、4014……P¥70,000
究極の万能ターミナルエミュレーターグラフィック可
Smarterm 100 (Persoft) VT100、VT102、VT152……P¥30,000
TOPS (Sun) PCs、Mac、Sun、UNIX のミックスでネット、Ethernet、Token サポート、Mac用も有り、デスク、ファイルのサーバー、異なるOS間でのやり取り、32台迄……¥33,000
Novell ELS Netware Level 11 8台……¥260,000
Novell Advanced Netware V2.15 100台……¥520,000

●OS
Unix System V/AT system (Microport)……U¥145,000
Marge286 (Microport sys)……U¥60,000
PC DOS V3.3 (IBM) Basica、3.5" ディスク付……P¥20,800
MS DOS V3.3 (Microsoft) GW Basic付……M¥29,000
OS/2 (IBM) マルチタスク、新製品 V1.1……P¥52,000
PC-DOS V4 (IBM) 新製品……P¥24,000
IBM OS/2 Programming Tool Kit V.1.1……P¥120,000
OS/2 Extendid Edition 1.0……¥127,200

●386関連ソフト(ハードウエアに80386が必要)
Phar Lap 386 Developer Kit (Phar Lap Software)……P¥115,000
ASM、Link、MiniBug、Run386を含む
Softguard386 Developer Kit……P¥220,000
NDP Windows (MicroWay) NDP C 386用ソース(¥50,000)……¥23,000
NDP C 386 (MicroWay) 32bit用……¥125,000
High C compiler 386 (Mete Ware) 386用……P¥189,000
Cコンパイラ、Run386か必要
PC-MOS/386 (Software Link) シングルユーザ……P¥60,000
5ユーザ(¥164,000)、25ユーザあり
Professional Pascal (Meta Ware)……P¥136,000
High C Compiler286……¥136,000

●ユーティリティ
BASTOC (JMI) MS BASIC をCに変身……P¥112,000
Norton Utilities Advance (Norton) ディスクファイル……P¥24,000
DIR修復、その他
FANCY FONT (Softcraft) プリンター用フォント……P¥48,000
PC TOOLS Deluxe (Centralpoint) 含む Fastback、Norton、XTREE、Mace、Sidekick、Diskopt……P¥17,000

●CAD、グラフィック
Interactive Easy Flow (Haven Tree Software)……P¥45,000
フローチャート作成CAD
smARTWORK (inter) アートワーク作成ソフト……P¥210,000
Generic CADD 3.0……P¥20,000
建築、工業デザイン、電気、ドラフト、アート専用、etc
HALO-88 (Media Cybernetics) C、Fortran、Basic、Pascal用……P¥65,000
Generic 3D 3次元のモデリングが可能……P¥51,000
AUTO CAD (Auto Desk) Re.9……P¥580,000
Autosketch Enhanced (Auto Desk) STDより高速……P¥21,000
Design Cad (American Small)……P¥42,000
VERSA CAD 2D or VERSA CAD DESIGN……¥510,000
Harvard Graphics製品ビジネスグラフのベストセラー、ロータス、DBASEにリンク可……¥CALL
MICROGRAFX製品……¥CALL

●計算
Reduce (Northwest Computer) Rand社……M¥95,000
計算用ソフト HDが必要
Math CAD (Math soft)……P¥57,000
Mathematica 386/387……¥174,000
Mathematica 386/Weitek……¥227,000
SYGRAPH (Systat) 科学技術用専用グラフィック……P¥95,000
SYSTAT (Systat) 科学技術計算、統計用ソフト MAC用あり……P¥115,000
(両方価格 P¥152,000)

●ASYST
Asyst Modules 1.2.3……R¥280,000
1.2.3.4 も有り
Asyst Modules 1.2.4……P¥280,000
Asyst Modules 1.2……P¥230,000
Asystant……P¥74,000
Asystant Plus……P¥138,000

●その他の言語とユーティリティ
DESQview V2.2 ウィンドー、マルチデスク、286用(EMS4.0、Lotus、Wordperfectに最適)……¥24,000
DESQview 386 386専用 QEMM-386含む……¥35,000
QEMM-386 386PCでEMS 3.2、EMS 4.0をサポート……¥12,000
Carbon Copy……P¥32,000
Microsoft COBOL……P¥140,000
PLOTIT LOTUS用データの統計、グラフ化処理……P¥110,000
Auto Flow-C、Pascal フローチャート解説……¥57,000

●データベース
dBase III Plus (Ashton)……P¥90,000
R:BASE 5000……¥76,000
Symphony (Lotus)……P¥103,000
Infomix SQL……P¥148,000
Ventura (xerox) デスクトップパブリッシング……P¥120,000
Page Maker デスクトップパブリッシング……P¥110,000
Lotus 1.2.3……P¥70,000
dBASE 1V (Ashton) 新製品……P¥109,000
SQL Window (Gupita) 新製品……¥295,000

IBM•AX系パソコンハードウェア 三洋電機、正規代理店

●コンピュータ
PC AT (IBM) 8MHz 512KB 1.2MFDD、30MBHD……¥800,000
IBM PS/2 MD30、MD30-286、MD50、MD60、MD70、MD80、IBM全機種……¥CALL
LAPTOP、AX、AT、Compatible……¥CALL
MAXY-A20(A12)、MP286(三菱)、J3100 SGT (101、041)、J3100 GX、J3300 (30、50)、J3100SL (002、001、021) (東芝)……¥CALL
T5200、T5100、T3100e、T1600、T1200、T1100+、T1000(東芝)海外版……¥CALL
MBC-17JH40 (JH20、JF)、MBC17LTJF (JH) (三洋)……¥CALL

●ディスプレイ
Multi Sync (NEC) USAで人気No.1の新世代ディスプレイ……¥CALL
MultiSync plus (NEC) XL(1024×768)も有り 960×720……¥185,000
MultiSync II (NEC) 新製品……¥145,000
Multi 2A 800×600、14" 新製品……¥145,000
Multi 3D 1,024×768、14" 新製品……¥166,000

●ディスプレイボード for IBM
VEGA Deluxe (Video 7) USAで高い評価……¥63,000
640×350、640×480、EGA、CGA、ハーキュリーズモード自動切替、
EGA CARD 640×350、CGA、ハーキュリーズ……¥35,000
VEGA (Video 7) VGA、EGA、MDA、CGA……¥74,000
V-RAM VGA (VIDEO 7) 超高速、高解像度、1024×768……¥130,000
FastWrite VGA (VIDEO 7) 超高速、高解像度、800×600……¥90,000
EVERGRAPHIC DELUXE モノクロ、1024×704、720×348 Ventula、CAD……¥38,000

●テープバックアップ EVEREX
EXCEL STREAM 40Int、40MB FDD タイプ……¥90,000
EXCEL STREAM 60Int、60MB (TEAC ¥170,000)……¥188,000
EXCEL STREAM 60MB WANGTEK EXT……¥220,000
EXCEL STREAM 60MB TEAC EXT……¥186,000
EXCEL STREAM 60MB、WANGTEK (PS/2用)……¥250,000
EXCEL STREAM 125MB WANGTEK (PS/2用) EXT……¥320,000
EXCEL STREAM 125MB EXTERNAL AT用……¥284,000
XENIX、LINIXに最適
EXCEL STREAM 125MB WT AT用……¥253,000

●ハードディスク for AT 仕様Internal(外部用電源有ります)
ST225 20MB (65ms) (Seagate) ハーフサイズ……¥60,000
ST251 40MB (40ms) (Seagate) ハーフサイズ……¥80,000
ST4096 80MB (28ms) (Seagate) フルサイズ……¥140,000
40MB (28ms) (CDC) ハーフサイズ……¥150,000
72MB (25ms) (東芝) フルサイズ……¥250,000
145MB (25ms) (Maxtor) フルサイズ……¥390,000
WD1002 新製品、インターリーブ1:2、高速……¥33,000
TMC-841DNK (Future) SCSIコントローラ……¥39,000
TMC-881DNK (Future) SCSIコントローラ、FDDx2……¥63,000

●数値演算素子
80287-8 (8MHz)……¥45,000
80287-10 (10MHz)……¥72,000
80287 Turbo (12MHz) 287キット……¥120,000
(システムクロックに無関係に12MHz) 10MHzも有り
80387-16 (16MHz)……¥120,000
80387-20 (20MHz)……¥140,000
80387-25 (25MHz)……¥160,000

●GPIB インターフェイスカード for IBM
PC488 (CEC) DMA使用時転送スピード700K Bytes/秒……¥80,000
C、Fortran、Basic用ファンクション、サブルーチンおよびシリアルポートへの割付用ソフトウェア付

●その他
REFORMATTER 変換システム、8ドライブ他……¥CALL
IBM3740、CPM、DEC RT-11
SCANMAN (LOGITECH) 200DPI、MS Window、Ventura……¥70,000
QUADRAM社 全製品、お取扱い致します……¥CALL
AT拡張ケース8スロットル……¥340,000
Multiport (ARNET) マルチユーザー、4-8SERIAL PORT……¥CALL
コンカレントDOS、XENIX、PCMOS サポート
Hercules Network Card Plus TOPS (Apple Talk) ネットワーク、モノクロ……¥60,000
TOPS Flash Card Apple Talkより3倍高速……¥43,000
3.5" FDD (internal) 1.44MB、ドライバー付……¥39,000
Display Extended Cable & Buffer 各種……¥CALL

●データアクジション、データアナリシス
Data Transaction……¥CALL
アナログデバイス……¥CALL
Keithley……¥CALL

●オプションボード for IBM
2MB RAMカード (Suntek) EMS、TEISK、SPOOL、ソフト付……¥37,000
2.5MB RAMカード (OK) 512Kに使用可……¥24,000
2MB EMS RAMカード (OK)……¥28,000
HDD/FDD コントローラカード (ウエスタンデジタル)……¥35,000
RAM10000 (EVEREX) 10MB、XENIX、UNIX用 EMS メモリ……¥51,000
モノクロディスプレイカード ハーキュリーズコンパチブル……¥15,000
Microsoft Mouse シリアル (Bus、¥34,000) Autosketch付……¥34,000
PC Mouse シリアルソフト付……¥32,000
Logic Mouse シリアルソフト付……¥32,000
2 シリアル/パラレルカード……¥15,500
2 シリアル/パラレルカード(4 アドレス切替)……¥26,000
Inboard 386/PC OK、IBM PC、XT用 10倍高速……¥210,000
Inboard 386 OK、IBM AT用……¥250,000
LAPLINK (Traveling) 2つのコンピューターをつなぐ……¥30,000
Desk-link (Traveling) 2台のコンピューターでディスクとプリンターのサーバー……¥30,000
Above Board PLUS (intel) 512kB付、EMS、4.0、6MB迄、新製品、高速……¥110,000
Intel 製品……¥CALL

●レザープリンタ
HP Laserjet II (HP)……¥430,000

●グラフィックス専用ボード(AT&T)
AT&Tが開発したグラフィック専用ボード
TARGA8……¥380,000
TARGA16……¥550,000
TARGA24……¥550,000

●3Com・NOVELL Ethernet
Etherlink II……¥85,000
Etherlink Plus……¥150,000
NE 1000 (NOVELL)……¥76,000

●その他
LaserMaster ポストスクリプトカード……¥CALL
ARTIST (Control Systems)……¥CALL
IRMAlink FT/3270 (DCA)、IRMA2、IRMAcom、IRMAprint (DCA)……¥CALL
Tranceiver (3 Com)……¥52,000

P

〒108 港区芝4-5-12 三田ハイツ2F
☎(03)455-3460㈹ FAX(03)457-7970
(JR田町駅三田口歩5分 営業時間9:00〜18:00 日曜、祭日、第1・2・3土曜定休 土曜12:00迄)
※全ての商品は日本信販の分割払い、リースを御利用できます。

8bitに帰れ！FA/LA/自作派のためのZ80クロスツール3本。

世の中すっかり16/32bit CPUがあたりまえの時代になりました。しかし、ちょっとした用途には8bitの方がとっつきやすいし、コスト的にも有利。でも開発はMS-DOS上でやりたいなとお考えのあなたに珠玉の三本。いずれもPC9801専用ですのでご注意下さい。

Z-VISION Z80フルスクリーンクロスデバッガ

SYSTEMLOAD ¥19,800

PC9801のMS-DOS上で動作する強力なZ80クロスデバッガです。ブレークポイント、レジスタ、ダンプリスト、ダイアログ、トレース情報を分割した画面上で一度に見ながら作業が行なえます。しかも、超低価格。

■割込みにも完全対応。モード2割込みもシミュレートできます。また、キー入力による割込みの他に仮想的なインターバルタイマを割込み源とすることができます
■入出力ポートをディスク上のファイルに割合てることができます。予め、入力データをテキストエディタで作ったり、出力結果をリストアップしたりが簡単に可能です
■コマンド入力のリダイレクトも可。■CP/M-80エミュレーション機能付

ZASM/ZZ マクロアセンブラ/ディスアセンブラ

プロテック・サービス 各¥10,000

PC9801のMS-DOS上で動作するターボ感覚のウィンドウベースZ80/64180クロスマクロアセンブラとクロスディスアセンブラです。
T.C2.0の全ソースリスト付！

■アセンブラ———リンカー内蔵。リロケータブルアセンブラです。マクロはMACRO-80に上位準拠。強力なMAKE機能をサポート。
■ディスアセンブラ—自動モードで強力なソースコードジェネレータに！ウィンドウを開いてクロス、リファレンス、未定義ラベル等を参照できます。ユーザータグをセット可能。作業中の状態を保存できます。

ADO·TOYOMURA T·ZONE ティー・ゾーン

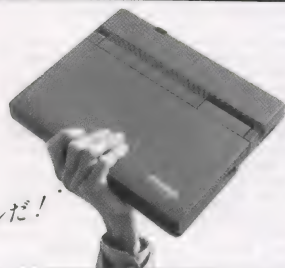
Micom Zone

2F 〒101 東京都千代田区外神田4-4-1 ☎257-2650

■すごい。■J3100SLとほぼ同機能でありながら、片手で(しかも水平に)持てる。■ノート型でありながらバックライトLCDを採用。■バッテリーで約2.5時間動作。しかも交換可能なパック方式。■これがなきゃ絶対イヤ／というほど使いやすいレジューム機能。電源ON／OFFは気軽にできなきゃネッ。■昔の人は言いました。コンパクトなキーボードは打ちやすい。リターンキーにも手が届くゾ／■フロッピーベースでPC9801とデータ交換可能。■そして何しろ安い¥198,000

TOSHIBA

神をも恐れぬネーミング?! その名は

フレームバッファ 無

ウェーブトレイン エスキース 定価¥23,000

電子水彩

フレームバッファは高くて手が出ない？
そんな方に朗報です。
エスキースがあなたの夢を実現します。

●毛筆タイプの筆を選べば、書道や墨絵のぼかしができます
●パレットナイフで、上に乗った色を削ったり乗せたりできます
●クレパスを指でこすって出すのと同じ、やわらかで微妙な効果を出せます
●透明度や密度を自在に組み合わせて、1つの筆にバリエーションをつけられます
●製図機能で、作った図の変形、移動、特殊な画像処理ができます

フレームバッファ 有

サピエンス スーパータブロー 定価¥58,000

1670万色フルカラーペイント

スーパータブロは、業界標準フルカラーフレームメモリ「スーパーフレーム2」上で走行するフルカラーペイント＆デザインシミュレーションソフトウェアです。スーパータブローは、パソコンの画面とスーパーフレーム2のフルカラー画面とがスーパーインポーズするという機能を縦横に使いこなし、素早いマウスカーソールの応答やマスク機能などを実現した対話型ソフトウェアです。

機能

●簡単な色指定とインクポット●豊富なブラシ●グラッドスクウェア●ルーペ●リアルタイム座標表示●ドラフト機能●画像の拡大、縮小、回転、変形●画像の拡大エディット●画像のFix/UNDO●マスクの基本プロセス●クイックトモグラフとオートマスク●マスクの演算●簡単な文字入力●従来の画像のフルカラー化●カラーコレクション●スクリーントーン機能●画像の合成

※このソフトを動かすにはスーパーフレーム(¥158,000)が必要です。店頭デモ中！

『安心料』がお安くなりました。

MPS-500 ハイコストパフォーマンス無停電電源装置

サンケン電気 定価¥39,800

パソコン用スイッチング電源供給のトップシェアを誇るサンケンがその豊富な経験を結集してパソコン専用無停電電源を開発しました。画期的な価格に注目！メーカー保証動作時間は負荷300W時約3分間ですが、コンピュータ・CRT・ハードディスクの標準的組合せで10分以上OKです。アラーム出力端子付です。(割込みによるデータの自動退避等の応用が可能です)

T·ZONE正社員・長期アルバイト募集中！
☆お問い合わせは総務課鈴木まで (TEL 03-257-2630)

P

T·ZONE

営業時間：AM10:30〜PM7:00

下記T·ZONE各店でも扱っています。

宇都宮店：☎0286(63)4949 川口店：☎0482(68)7826 ラジオショップ：☎03(257)2643 横浜店：☎045(641)7741
大宮店：☎048(652)1831 東ラジ店：☎03(257)2694 パーツショップ：☎03(257)2655 静岡店：☎0542(83)1331

●マイコン通販利用の方へ：現金書留で送金される際は、住所、氏名、TEL番号、希望商品名(詳しく)を明記して下さい。振込を御希望の方は下記銀行へお願いします。尚、いずれも予めTELにて、御予約・送料確認の上御送金下さい。(振込口座 埼玉銀行 秋葉原支店 当座2705 株亜土電子工業)

どれが お好み？

〔☆LANGUAGE〕

Product	Price
APL PLUS/PC V8.0	¥118,000
ASM FLOW (FOR ASEMBLER)	¥18,000
BTRIEVE	¥85,000
BTRIEVE/N	¥200,000
C-TERP	¥26,800
C/UTILITIES TOOLCHEST	¥6,800
DBC-III LIBRARY (FOR LATTICE C)	¥38,000
DBCIII PLUS (FOR LATTICE C)	¥90,000
DOUBLE DOS 5.0	¥10,800
DSD87	¥20,000
HDC WINDOWS EXPRESS V2.1	¥14,800
INSIDE' (FOR COMPILER)	¥13,800
LATTICE C COMPILER FOR DOS & OS/2	¥55,800
LOGITECH MODULA-2 COMPILER	¥16,800
LOGITECH MODULA-2 DEVELOPMENT SYSTEM	¥43,800
LOGITECH MODULA-2 OS/2	¥60,000
LOGITECH MODULA-2 TOOLKIT	¥29,800
LOGITECH MODULA-2 WINDOW PACKAGE	¥8,800
MICROSOFT BASIC COMPILER V6.0 (DOS/OS2)	¥48,000
MICROSOFT BASIC INTERPRETER V5.28	¥58,000
MICROSOFT C5.1 OPTIMIZING COMPILER	¥74,800
MICROSOFT COBOL COMPILER V3.0 (DOS/OS2)	¥148,000
MICROSOFT FORTRAN COMPILER V4.1 (DOS/OS2)	¥69,800
MICROSOFT MACRO ASEMMBLER V5.1	¥23,800
MICROSOFT PASCAL COMPILER V4.0 (DOS/OS2)	¥48,800
MICROSOFT PROGRAMMERS OS/2 TOOLKIT	¥56,800
MICROSOFT QUICK BASIC V4.5	¥16,000
MICROSOFT QUICK C V2	¥16,000
MICROSOFT QUICK PASCAL V1.0	¥17,800
MS-DOS V4.01 (W/GW-BASIC)	¥25,000
OPTASM	¥20,000
OPTDEBUG	¥20,000
OPTLIB	¥9,000
OPTLINK	¥20,000
OS/2 EXTEND EDITON V1.1	¥168,000
OS/2 STANDARD EDITION V1.1	¥68,000
PC-DOS V3.3	¥23,700
PC-DOS V4.0	¥30,000
PC-METRIC (FOR LANGUAGE)	¥32,000
PERSONAL REXX	¥30,000
PLINK86 PLUS	¥68,000
POLYAWK V1.2	¥16,800
POLYLIBRARIAN V1.3B	¥18,000
POWER C	¥6,800
POWER C TRACE DEBUGGER	¥6,800
PULE MASTER 3 (EXPERT SYSTEM)	¥90,000
SMALL TALK V	¥18,000
SOURCER	¥20,000
TOPSPEED MODULA-2 DOS 3-PACK	¥35,800
TOPSPEED MODULA-2 DOS COMPILER	¥18,800
TOPSPEED MODULA-2 TECHKIT	¥11,800
TOPSPEED MODULA-2 VID	¥11,800
TURBO ASSEMBLER/DEBUGGER	¥22,000
TURBO BASIC	¥14,800
TURBO BASIC DATABASE TOOLBOX 1.0	¥16,000
TURBO BASIC EDITOR TOOLBOX 1.0	¥15,800
TURBO C PRO PACK	¥38,000
TURBO C V2.0	¥22,000
TURBO PASCAL EDITOR TOOLBOX 4.0	¥15,800
TURBO PASCAL TUTOR V4.0	¥9,800
TURBO PASCAL V5.0	¥22,000
TURBO PASCAL V5.0 PRO PACK	¥38,000
TURBO PASCAL V5.5	¥27,000
TURBO PASCAL V5.5 PRO PACK	¥43,000
TURBO PROLOG TOOLBOX	¥14,800
TURBO PROLOG V2.0	¥22,000
VM/386	¥45,800
VP EXPERT V2.0	¥33,800
ZORTECH C COMPILER & DEBUGGER	¥29,800
ZORTECH C VIDEO SET	¥63,800
ZORTECH C++ COMPILER	¥29,800
ZORTECH C++ TOOLS	¥24,800

〔☆INTEGRATIVE SOFTWARE〕

Product	Price
101 MACROS PLUS FOR EXCEL	¥10,800
ALPHA WORKS V1.0	¥29,800
ENABLE OA (DOS & OS/2)	¥79,800
FRAMEWORK III V1.0	¥105,000
MICROSOFT WORKS V1.05	¥22,800
PFS:FIRST CHOICE 3.0	¥19,800
Q & A V3.0	¥47,800
SMARTSYSTEM V3.1	¥99,800
SYMPHONY	

〔☆DATABASE〕

Product	Price
101 UTILITIES FOR dBASEIII PLUS	¥10,800
@BASE (FOR 1-2-3)	¥29,800
ALPHA/FOUR V1.0	
CLLIPER	¥108,000
DATAEASE V4.0	
DBASEIII PLUS V1.1	¥79,800
DBASE IV	¥118,000
DBFAST/DOS	¥16,800
DBXL (DIAMOND REL) V1.2	¥29,800
DBXL/LAN	¥88,000
FOXBASE PLUS DEVELOPMENT V2.10	¥48,000
FOXBASE PLUS/386	
PARADOX OS/2	¥98,000
PARADOX V2.0	¥98,000
PARADOX V3.0	¥128,000
PROFESSIONAL FILE V2.0	¥36,800
QUICKSILVER	¥99,800
R:BASE FOR DOS V2.1	¥89,800
RAPIDFILE V1.2	¥45,800
REFLEX ANALYST AND WORKSHOP	¥29,800
REFLEX V1.14	¥22,000
REFLEX V2.0	¥42,800
REFLEX WORKSHOP	¥10,500
VP INFO	¥20,000

〔☆SPREADSHEETS〕

Product	Price
101 MACROS PLUS FOR LOTUS 1-2-3	¥10,800
ALLWAYS (FOR 1-2-3)	¥22,800
INWORD (FOR 1-2-3)	¥14,800
LOOK & LINK (FOR 1-2-3)	¥14,800
LOTUS 1-2-3	¥99,000
LUCID 3-D V2.0	¥14,800
MICROSOFT MULTIPLAN V4.0	¥29,800
NOTEWORTHY (FOR 1-2-3 OR SYMPHONY)	¥12,800
PFS:PROFESSIONAL PLAN	¥13,800
QUATRRO 1.0	¥35,000
SIDEWAYS V3.2 (FOR 1-2-3 AND SYMPHONY)	¥9,800
SILK V1.0	¥17,800
SUPERCALC4（在庫限り）	¥59,000
SUPERCALC5	¥78,800
VP PLANNER PLUS V2.0	¥33,800
WORKSHEET UTILITIES (FOR 1-2-3)	¥14,800

〔☆WORD PROCESSING〕

Product	Price
BRIEF V2.1	¥38.000
DBRIEF (FOR BRIEF 2.1)	¥18,000
EW+	¥95.000
GEM/3 FIRST WORD PLUS	¥17,800
KEDIT	¥33.800
LOTUS HAL	
MANUSCRIPT	
MICROSOFT WORD V4.0	¥39.800
MICROSOFT WORD V5.0	¥69.800

PC EXCEL

Product	Price
MULTIMATE ADVANTAGE II (PREMIUM PACK)	
MULTIMATE ADVANTAGE II V1.0	¥59.800
PROFESSIONAL WRITE V2.0	¥25.800
RIGHTWRITER V3.0	¥12.800
SPRINT 1.0	¥29.800
THE PERFECT ADDITION (FOR WORDPERFECT)	¥6.800
TURBO LIGHTNING	¥14.800
VEDIT PLUS	¥33.800
WEBSTER'S NEW WORLD SPELLING CHECKER	¥9.800
WORD PERFECT LIBRARY V2.0	¥17.800
WORD PERFECT V5.0	¥57.800
WORDSTAR PROFESSIONAL REL 5.0	¥59.800
WORDSTAR2000 PLUS REL.3 PERSONAL	¥59.800

〔☆DESKTOP PUBLISHING〕

Product	Price
BYLINE	¥44.800
CERTIFICATE MAKER	¥5.000
GEM/3 DESKTOP PUBLISHER V2.0	¥43.800
PAGEMAKER 3.0	¥115.000
PFS:FIRST PUBLISHER V2.0	¥17.800
PIZAZZ PLUS V1.2	¥25.800
SCORE V3.1	¥135.000
VENTURA PUBLISHER PROFESSIONAL EXTENSION	¥88.000
VENTURA PUBLISHER V2.0	¥99.800

〔☆GRAPHICS〕

Product	Price
3-D GRAPHICS V1.02	¥29.800
BANK STREET WRITER PLUS	¥11.800
ESSENTIAL GRAPHICS V2.0 (FOR C, FORTRAN)	¥54.800
FREELANCE PLUS	
GEM/3 DRAW PLUS	¥43.800
GRAPHIC	¥72.000
HARVARD GRAPHICS V2.1	¥59.800
MANAGING YOUR MONEY 4.0	¥25.000
MICROGRAFX DESIGNER V2.0	¥120.000
MICROSOFT CHART V3.0	¥59.800
NEWSMASTER II	¥9.800
PINSTRIPE PRESENTER	¥29.800
PRINT SHOP	¥7.800
PRINT SHOP GRAPH. LIB DISK 1	¥5.000
PRINT SHOP GRAPH. LIB DISK 2	¥5.000
PRINT SHOP GRAPH. LIB HOLLDAY DISK	¥5.000
PUBLISHER'S PAINTBRUSH V1.6	¥36.800
STAT GRAPHICS V3.0	¥148.000

〔☆COMMUNICATIONS〕

Product	Price
BROOKLYN BRIDGE SERIAL	¥19.800
CARBON COPY PLUS V5.0	¥27.800
CROSSTALK MK.4 V1.01	¥29.800
CROSSTALK X VI V3.61	¥25.800
LAP-LINK PLUS	¥17.800
LAPLINK III	¥20.000
MIRROR III	¥15.800
PROCCOMM PLUS	¥13.800
PROFESSIONAL CONNECTION V3.0	¥6.800
RELAY GOLD V3.0	¥39.800
RELAY SILVER V1.0	¥24.800
SIDE TALK	¥18.800
SMARTCOM II (3.0)	¥23.800
SMARTCOM III V1.0	¥35.800

〔☆CAD/DESIGN〕

Product	Price
AUTOSKETCH ENCHANCED V1.04	¥16.800
AUTOSKETCH STANDARD V1.04	¥13.800
CADKEY V3.02	
CORELDRAW	¥88.000
DELUXE PAINT II	¥12.800
DESIGNCAD 3-D V1.1	¥39.800
DESIGNCAD 3-D V2.0	¥52.800
DESIGNCAD V3.1 (NEW PRODESIGN)	¥39.800
EASY CAD2 V2.05	¥25.800
GENERIC 3D SOLIDS V3.0	
GENERIC CADD LEVEL1	¥7.800
GENERIC CADD LEVEL2	¥21.000
GENERIC CADD LEVEL3	¥43.800
GENERIC CADD STARTER KIT	¥25.800
MATHCAD V2.5	¥55.800

Product	Price
MICROCAP III ANALOG V2	¥270.000
MICROLOGIC II	¥130.000
PADS-CAE II	¥180.000
PADS-PCB	¥360.000
PADS-PCB+ROUTE	¥594.000
PADS-PCB+ROUTE+LARGE	¥706.500
PADS-SUPERROUTER	¥108.000
SCHEMA（回路図CAD）	¥180.000

〔☆LEARNING〕

Product	Price
ALGE-BLASTER!	¥7.800
DS TUTOR	¥6.800
LEARN TO USE YOUR PC	¥5.800
MATH BLASTER PLUS!	¥6.800
MAVIS BEACON TEACHES TYPING GRAMMATIK III	¥7.800
MICROSOFT LEARNING DOS V2.0	¥8.800
PC INSTRUCTOR	¥7.800
PROFESSOR DOS	¥8.800
QUICKEN V2.1	¥6.800
READER RABBIT	¥4.800
SPEED READING TUTOR IV	¥7.800
TRAINING FOR LOTUS 1-2-3	¥11.800
TRAINING FOR WORDPERFECT V5.0	¥11.800
TRAINING FOR dBASE III	¥11.800
TRAINING FOR dBASE IV	¥11.800
TYPING TUTOR IV	¥7.800

〔☆GAMES〕

Product	Price
50 ANNOTATED CHESS CLASSICS	¥4.800
ACE ≠2	¥6.800
ACE OF ACES	¥7.800

TETRIS

Product	Price
ANCIENT ART OF WAR	¥7.800
ANCIENT ART OF WAR AT SEA	¥8.800
B-24	¥7.800
BATTLE CHESS	¥9.800
CALIFORNIA GAMES	¥5.800
CHAMPIONSHIP BASEBALL	¥8.800
CHAMPIONSHIP GOLF	¥8.800
CHAMPIONSHIP BACKGAMMON	¥4.800
CHAMPIONSHIP BOXING	¥6.800
CHAMPIONSHIP LODE RUNNER	¥6.800
CHESSMASTER 2000	¥8.800
CHUCK YEAGER'S ADVANCED FLIGHT SIMULATOR	¥6.800
CONFLICT IN VETNAM	¥7.800
CORRUPTION	¥7.800
DARK CASTLE	¥8.800
DEEP SPACE OPERATION COPERNICUS	¥3.800
DIVE BOMBER	¥7.800
DR. J & LARRY BIRD GO ONE ON ONE	¥3.500
F-15 STRIKE EAGLE	¥5.800
F-19 STEALTH FIGHTER	¥10.800
FALCON	¥7.800
FALCON AT	¥9.800
GBA CHAMPIONSHIP BASKETBALL	¥8.800
GFL CHAMPIONSHIP FOOTBALL	¥8.800
GOLF'S BEST:PINEHURST II	¥5.800
GOLF'S BEST ST. ANDREWS	¥5.800
GUERILLA WAR	¥7.800
GUNSHIP	¥9.800
HACKER II	¥8.800
HARDBALL	¥3.000
HELICOPTER SIMULATOR	¥7.800
JET FLIGHT SIMULATOR W/JAPAN DISK	¥11.800
JORDAN VS. BIRD	¥5.800
KARATEKA	¥6.800
KING'S QUEST I	¥8.800
KING'S QUEST II	¥8.800
KING'S QUEST III	¥8.800
KING'S QUEST IV	¥8.800
LEISURE SUIT LARRY	¥6.800
LEISURE SUIT LARRY II	¥8.800
LODE RUNNER	¥6.800
LUNAR EXPLORER	¥6.800
MEAN 18, ULTIMATE GOLF	¥6.800
MICROSOFT FLIGHT SIMULATOR V3.0	¥7.800
MIGHT AND MAGIC	¥10.800
NFL CHALLENGE	¥17.800
ORBITER	¥8.800
PATTON VS. ROMMEL	¥8.800
PETE ROSE PENNANT FEVER	¥8.800
PHARAO'S REVENGE	¥5.800
PINBALL CONSTRUCTION SET	¥3.500
PINBALL WIZARD	¥6.800
PIRATES!	¥6.800
PLATOON	¥7.800
POLICE QUEST II	¥8.800
PT109	¥6.800
ROCKET RANGER	¥9.800
SARGON III	¥4.800
SHANGHAI	¥7.800
SOKO-BAN	¥6.800
SPACE QUEST II	¥8.800
STAR FLIGHT	¥8.800
STAR SAGA I	¥15.800
STAR TREK	¥8.800
STAR TREK II	¥8.800
STAR TREK:REBEL UNIVERSE	¥8.800
SUPERBIKE CHALLENGE	¥4.000

WORDSTAR PROFESSIONAL

Product	Price
TEST DRIVE	¥7.800
TETRIS	¥5.800
THE GAMES SUMMER EDITION	¥8.800
THE KING OF CHICAGO	¥9.800
THE SEVEN SPRITS OF RA	¥4.000
THE THREE STOOGES	¥9.800
THE UNIV. MILITARY SIMULATOR	¥9.800
THE USURPER MINES OF QYNTARR	¥5.800
THE WIZARD OF OZ	¥3.800
THE WORLD'S GREATEST BASEBALL	¥4.800
THEXDER	¥5.800
VICTORY ROAD	¥7.800
WHERE IN THE USA IS CARMEN SANDIEGO	¥6.800
WHERE IN THE WORLD IS CARMEN SANDIEGO	¥6.800
WILDERNESS	¥6.800
WINTER GAMES	¥8.800
WIZARDRY 1-PROVING GROUNDS	¥11.800
WIZARDRY 2-KNIGHT OF DIAMONDS	¥9.800
WIZARDRY 3-LEGACY OF LYLGAMYN	¥9.800
WIZARDRY 4-THE RETURN OF WERDNA	¥11.800
WIZARDRY 5-HEART OF THE MAELSTROM	¥9.800
WORLD GAMES	¥8.800
WORLD TOUR GOLF	¥10.800

〔☆PERSONAL INFO. MANAGER〕

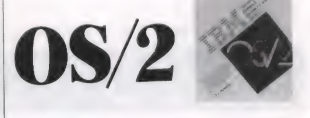
Product	Price
ASKSAM V4.1	¥49.800
GRANDVIEW V1.0	¥45.800
LOTUS AGENDA	
MICROSOFT PROJECT V4.0	¥78.000
TIMELINE V3.0	¥78.000

〔☆LAN〕

Product	Price
TOPS FOR DOS V2.1	¥29.800
MUSIC CONSTRUCTION SET	¥3.800

〔☆UTILITIES〕

Product	Price
BOOKMARK	¥12.800
BOOKMARK PLUS	¥28.000
C-DISPLAY MANAGER (FOR C)	¥24.000
CLEAR FOR dBASE	
COCOON V1.0	¥12.800
COPY II PC DELUXE OPTIONBOAD	¥29.800
COPY II PC V5	¥6.800
COREFAST V2.0	¥24.800
CUBIT V2.0 (FILECOMPRESSION)	¥10.800
DAN BRICKLIN'S DEMO II	¥29.800
DBASE PROGRAMMER'S UTILITIES	¥13.800
DBASE PROGRAMMER'S UTILITIES VII	¥13.800
DESQVIEW 386	¥28.000
DESQVIEW V2.2	¥19.800
DISK OPTIMIZER WITH DATA GURDIAN V4.0	¥10.800
DISK TECHNICIAN	¥16.800
DISK TECHNICIAN ADVANCED	¥29.800
DISK TECHNICIAN PLUS	¥19.800
DOS POWER TOOLS	
EASYFLOW	¥28.800
EUREKA THE SOLVER	¥25.000
FASTBACK PLUS V2.0	¥25.800
FLASH V6.0	¥11.800
FLOW CHARTING II	¥38.800
HYPERPAD	¥17.800
LIGHTNING WORD WIZARD	
MACE GOLD	¥20.800
MACE UTILITIES V5	¥16.800
MAGIC MIRROR	¥15.800
MEMORYMATE V3.0	¥10.800
NORTON COMMANDER V2.0	¥12.800
NORTON EDITOR	¥10.800
NORTON GUIDES:C	¥14.800
NORTON UTILITIES ADVANCED V4.5	¥19.800
NORTON UTILITIES V4.5	¥13.800
OMNIVIEW	¥16.000
PC TOOLS DELUXE V5.1	¥11.800
PC TOOLS DELUXE V5.5	¥21.800
PC-FULLBAK	¥8.800
POLYMAKE V3.0	¥27.000
POPDROP V3.1	¥6.800

OS/2

Product	Price
PRINT Q V4.0	¥19.800
PVCS CORPORATE V2.1B	¥72.000
PVCS PERSONAL V2.1B	¥27.000
QUICK READER V1.5 R2.0	¥55.000
REFREE	¥9.800
SEE MORE V1.02 (FOR 1-2-3)	¥11.800
SEE MORE V1.02 (FOR SYMPHONY)	¥14.800
SIDEKICK PLUS V1.0	¥32.800
SIDEKICK V1.5	¥13.800
SMARTNOTES V2.0	¥11.800
SOFTSAFE	¥12.800
SOFTWARE CAROUSEL V2.0	¥11.800
SORT FACILITY (MS-DOS) V1.01	¥34.800
STATCHECK V3.0	¥16.800
SUPER PC-KWIK POWER PAK	¥22.800
SUPER PC-KWIK V3.08	¥15.800
SUPERKEY V1.16	¥15.800
TAKE TWO MANAGER V2.1	¥22.800
V-EMM	¥17.800
XTREE PROFESSIONAL V1.0	¥17.800
XTREE V2.0	¥9.800

〔★BUSINESS〕

Product	Price
FORM FILER V2.0	¥18.800
FORM TOOL V2.01	¥11.800
PERSONAL LAWYER V2.0	¥7.800
SFT NETWARE V2.1	

〔☆WINDOWS〕

Product	Price
DBFAST/WINDOWS	¥48.000
GRAPH PLUS	
HDC WINDOWS COLOR	¥9.800
MICROSOFT PC EXCEL V2.1	¥75.000
MICROSOFT WINDOWS SOFTWARE DEV KIT V2.1	¥80.000
MICROSOFT WINDOWS286 V2.1	¥14.800
MICROSOFT WINDOWS386 V2.1	¥27.800

〔☆386 PRODUCTS〕

Product	Price
386-TO-THE-MAX V4.0	
386:ASM/386:LINK	¥80.000
386:DEBUGGER	¥32.000
386:DOS-EXTENDER	¥248.000
PARADOX 386	¥120.000
PC-MOS/386	¥34.800
QEMM-386 V4.2 (EXPANDED MEMORY MANAGER)	¥10.000

〔☆CD-ROM〕

Product	Price
MICROSOFT BOOKSHELF (CD ROM)	¥52.000
MICROSOFT PROGRAMMER'S LIBRARY (CD ROM)	¥70.000
MICROSOFT STAT PACK (CD ROM)	¥28.000
QUICK ART (CD ROM)	

◉IBM PC/AT用ソフト＆ハード、AXソフト＆ハード、PDS各種取り扱い。

AX・輸入ソフトの専門ショップ

プラス・ワン PLUS ONE

■Fox BASE DEBEROPMENT V2.1
Fox SOFTWARE／¥77,000-

1987, 1988年PCマガジン編集部特選獲得／
卓越したスピードと、100%dBASEコンパチブルですぐれた安定性、発展性をもつ環境ソフトです。
バージョン2.10ではデータベースの仕事をなめらかにこなし、新しい特徴としては、プルダウン、ポップアップメニュー、スクリーンペインター、アプリケーションジェネレータ、テンプレート・ランゲェジ・システム、オートマチックプログラムドクメンターがあります。
Fox BASE＋はローカルエリアネットワーク及び386マシン用のバージョンもあります。

■VP-Planner Plus
PAPERBACK SOFTWARE／¥48,000-

キーストローク、ファイル、マクロが1-2-3とコンパチブルなスプレッドシートプログラムです。dBASEファイルに、スムースで柔軟性のあるアクセスができ、“what-if”アナリシス、undo/redoコマンド、簡単なマクロコーディング、レポート作成の為のテキストエディットコマンド、バックグラウンド、再計算、そしてマルチディメンション機能を新しく組み込んだデータベースです。

■Lucid 3-D Version 2.0
DAC ／¥21,900-

1987年PCマガジンテクニカルエクセレンス賞獲得／
1988年Home Officeコンピュータエディタピックス賞獲得／
多方面で高い評価を受けている手頃な価格のスプレットシートです。3Dグラフィックス、ロータスのファイルの読み書き、セル間での3次元計算、無限のスプレッドシートサイズ、その他多くの機能を持っています。

■SPRINT
BORLAND INTERNATIONAL／¥47,000-

スプリントは、あなたをワープロの達人にしてしまうトータルな柔軟性をもった欧文ワードプロセッサです。

驚くべき速さ
スクロールや、文章の切り替えを一瞬にして行ない、ショットカットキーによりすばやい反応で簡単に編集ができます。

うれしい機能
スペルチェック機能により、入力した文章のチェックが簡単に行なう事ができます。また、自動的に文章をセーブする機能など他にもうれしい機能がついています。

非常に高い柔軟性
オプショナルユーザーインターフェースにより、他のワープロの操作に慣れている方はその操作感覚が得られ、生産能力が高められます。
ポップアップメニュー、ファンクションキーが選択自由で、しかも独自のメニューでバイパスキーを設定できるスタンダードインターフェースを備えています。

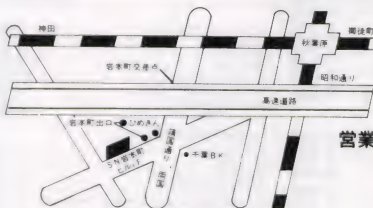
プロフェッショナル文章作成
スプリントは、プリンタがサポートできるフォントやタイプセットを自由に選択でき、また、それをいつでも変更できます。ジャスティフィケーションのための、両倒な計算をする必要はありません。さらにポストスクリプトの使用で文章をより美しく飾ることもできます。
長い文章もスプリントに内蔵された見出しと、自動のインデックス、目次、さらにはダイナミックなクロスリファレンスでより簡単に作成できます。
スプリントはHPのレーザージェットや植字機(タイプセッター)を含む300機種以上のプリンタをサポートしています。

ソフトウェアの世界博プラス・ワン

プラス・ワンは、新たなコンピュータシーンを展開するNew shopです。

■言語ソフト			
BASIC COMPILER V6.0(DOS/OS2)	MICROSOFT CORPORATION	5	¥ 80,600
C COMPILER V5.1(DOS/OS2)	MICROSOFT CORPORATION	5	¥ 123,000
COBOL COMPILER V3.0(DOS/OS2)	MICROSOFT CORPORATION	5	¥ 246,100
FORTRAN COMPILER V4.1(DOS/OS2)	MICROSOFT CORPORATION	5	¥ 123,000
Lahey PERSONAL FORTRAN 77	LAHEY COMPURTER	5	¥ 36,000
MICROSOFT QUICK C COMPILER V2	MICROSOFT CORPORATION	5&3	¥ 28,600
MODULA-2 COMPILER	LOGITECH. INC.	5	¥ 22,280
POWER C COMPILER	MIX SOFTWARE	5&3	¥ 8,000
SMALLTALK/V	DIGITALK INC.	5	¥ 33,600
■OS関連ソフト			
MICROSOFT MS-DOS Ver. 4.01	MICROSOFT CORPORATION	5	¥ 43,100
MICROSOFT WINDOWS 286 V2.1	MICROSOFT CORPORATION	5&3	¥ 23,000
MICROSOFT WINDOWS 386 V2.1	MICROSOFT CORPORATION	5&3	¥ 43,200
■表計算ソフト			
1-2-3 V2.01 W/VALUE PACK	LOTUS	5&3	¥ 110,800
MULTIPLAN V4,01	MICROSOFT CORPORATION	5&3	¥ 48,400
PFS:PROFESSIONAL PLAN	SOFTWARE PUBLISHING CORP	5&3	¥ 21,100
■データベース			
FOXBASE+DEVELOPMENT V2.1	FOX SOFTWARE	5&3	¥ 77,000
■統合ソフト			
FRAMEWORK III	ASHTON-TATE	5&3	¥ 160,600
MICROSOFT WORKS V1.05	MICROSOFT CORPORATION	5&3	¥ 34,600
SYMPHONY V2.0	LOTUS	5&3	¥ 155,200
Q&A V3.0	SYMANTEC CORP	5	¥ 79,200
■AIシステム等の特殊なソフト			
PC-TEX	PERSONAL TEX	5	¥ 99,600
VP-EXPERT V2.0	PAPERBACK SOFTWARE	5	¥ 47,600
■その他ユーティリティ			
PC TOOLS DELUXE V5,1	CENTRAL POINT SOFTWARE	5&3	¥ 17,500
■ワードプロセッサ			
DAC EASY WORD V3.0	DAC	5	¥ 15,000
MICROSOFT WORD V5.0	MICROSOFT CORPORATION	5	¥ 105,600
MULTIMATE V3.31	ASHTON-TATE	5&3	¥ 92,400
NORTON EDITOR V1.3C	PETER NORTON COMPUTING	5&3	¥ 14,000
RIGHT WRITER V3.0	RIGHTSOFT, INC.	5&3	¥ 18,500
WORDSTAR PROFESSIONAL REL 5	MICROPRO INT.'L CORP	5&3	¥ 94,000
WORD PERFECT V5.0	WORDPERFECT CORP	5&3	¥ 85,800
■CADソフト			
DESIGNCAD 3D V2,1	AMERICAN SMALL BUSINESS	5	¥ 77,200
DESIGNCAD V3.0	AMERICAN SMALL BUSINESS	5	¥ 57,700
■グラフィックソフト			
HARVARD GRAPHICS V2.1	SOFTWARE PUBLISHING	5	¥ 99,800
■通信・ターミナルソフト			
CROSSTALK MK.4V1.1	DCA/CROSSTALK COM.	5	¥ 44,800
TOPS FOR DOS V2.1	SUN MICROSYSTEMS	5	¥ 38,800

※本価格は、平成元年7月1日現在のものです。消費税は含まれておりません。

■取り扱い商品／各社AXマシン・周辺機器・国内のAXマシン用ソフト・輸入ソフト(米国ランキング上位ソフト他常備)、各種米国パソコン誌最新号1000円均一(BYTE、PC-MAGZINE、PC-WORD、Dr.DOBBS、MAC USER、MAC WORD)その他、ご注文に応じて色々なソフトを輸入販売致します。
■IBM—PC用PDSソフト(多種)配布サービス実施中。
(手数料ディスケット1枚当たり1000円)

JR秋葉原駅
昭和通り口より徒歩5分
都営新宿線岩本町駅
(出口A-5)より徒歩1分

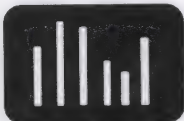
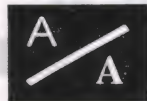
営業時間：10:00AM～7:00PM
(土・日・祭日を除く)

kansai 〒101 東京都千代田区岩本町3-2-10 SN岩本町ビル1F TEL.(03)5687-6905 FAX.(03)5687-6906

電脳経営軽量級挑戦者。

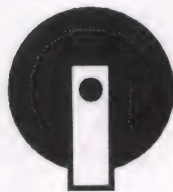
メモリ
●DEC社
●CLEAPOINT社

CRTターミナル
●DEC社
●伊藤忠

コミュニケーション
インタフェイス
●DEC社
●EXCELAN社

磁気テープ装置
●DEC社
●CIPHER社

同じ仕様のマシンならば、コストは低いほうがいい。軽量級とは低コストを指します。DEC社VAX11/PDP11に優れた周辺機器を組み合わせることにより高度のカスタマイズド・マシンを提供、挑戦する私たちは、経験豊かなエンジニアリングを発揮しています。

USED COMPUTER・SALE & RENT

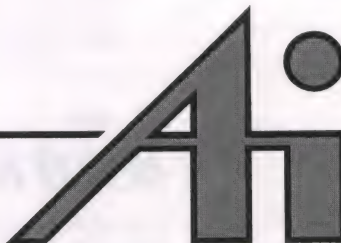
VAX11®/PDP11®

ディスク装置
●DEC社
●富士通

プリンタ
●DEC社
●東京重機

販売機種
●VAX11/785
●VAX11/780
●VAX11/750
●MICRO VAX Ⅱ
●PDP11/23
●PDP11/73
●PDP11/44

レンタル機器
●VAX11/785
●VAX11/780
●VAX11/750
●MICRO VAX Ⅱ
●PDP11/23
●PDP11/73
各種周辺機器

●ご希望のシステムを組立てます。
●ご注文から約1週間で納入いたします。
●上記以外にも各機種があります。お問い合せください。
●レンタル期間は問いません。

使用済のVAX/PDPを
高価にて買入いたします。

株式会社エイ・アイシステムズ　AI systems corporation

〒151 東京都渋谷区代々木5-67-5
Phone:03-468-6841
Fax　:03-468-6843

資料請求No.355

H8シリーズボードコンピュータ
HSB8/532-B

新世代の実力を秘めて
今、H8/532マイコン新登場

定価38,000円(表示価格には消費税は含まれておりません。)

H8/532は、日立オリジナル・アーキテクチャを採用した、内部16ビット構成の高速CPUを核にして、豊富な周辺機能を集積した高性能シングルチップ・マイクロコンピュータです。
また本ボードは、ソフト開発に欠かす事のできないデバッカをオプションとして準備しており、組込用としてユーザーのニーズにお応えできるものです。

写真のROM、RAMは、実際には、実装されてません

H8/532はワンタイムEPROM内蔵
マイコンのベストセラーZTAT®マイコンです

■仕様

- CPU:HD6475328CP10(PLCC)
- クロック:9.8304MHz(Xtal 19.6608MHz)
- メモリーアドレス:マキシマムモード、160Kバイト 空間1Mバイトに拡張
  ミニマムモード、64Kバイト 空間64Kバイト
- I/Oアドレス:128バイト、ユーザー解放80バイト(拡張ボード使用にて)
  ハードによるウエイト挿入回路使用
- メモリー:ROM 27256/62256 選択可――――1
  RAM 62256(バッテリーバックアップ可. 注)――――3
  メモリーはソケット仕様で実装されていません
- CPU内蔵:ROM 32Kバイト　RAM 1Kバイト
- 割込:NMI、IRQ0、IRQ1
- CPU内蔵:ADコンバータ 10ビット、8チャンネル、基準電源回路:HA17431P
  入力:オペアンプ、バッファ付き、変換時間 10MHz時1チャンネル当り13.8μs
- RS232C:2チャンネル　Ch1 CPU内蔵チャンネル
  Ch0 8251A(9600、4800、2400、1200BIT/S)
- パラレルI/O:P1 8255A、オール解放
- パラレルI/O:P2 CPU内蔵I/O PORT7 8ビット、PORT6 4ビット、PORT9 8ビット
- RTC:MSM6242、バッテリーバックアップ可. 注
- 使用I/OコネクターはMIL規格準拠品
- 基板寸法:160×115
- 電源:+5V(1A)、±12V(0.1A)、ユーザーで御用意下さい
  上記の他にCPU内蔵として次の機能が有ります
  ・16ビット、フリーランニングタイマ　・ウォッチ・ドッグドタイマ
  ・8ビットタイマ　・PWM、タイマ
  注)バッテリーは実装されてません

CPUナシ仕様:HSB8/532-B(CPUナシ)――――¥35,000
ソケットのみを実装LCC(窓付)用
(実装ソケット:3M 284-1273-00-1102J)

ROMライター アダプター
CPU内蔵ROMへ一般のROMライターで書き込む為のアダプターです。
LCC用……………定価15,000円

上記表示価格には消費税は含まれません

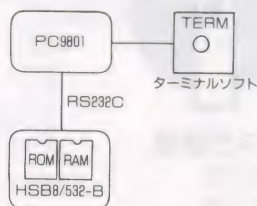
※ZTATは㈱日立製作所の登録商標です。

ICEROM8/532
デバッカ

ミニマムモード　定価25,000円
マキシマムモード 開発中 定価25,000円

※上記表示価格には消費税は含まれておりません。

本デバッカは北斗電子製ワンボードコンピュータHSB8/532-B用です。ボードにはデバッカの書込まれたROM(27256)とRAMを実装、モード64K空間をアクセスするミニマムモードと、最大1Mまでの空間をアクセスするマキシマムモードの2タイプが有ります。

〈構成〉

〈内容〉 ROM(デバッカ入)――――1
RAM(62256)――――1
ターミナルソフト TERM(2HD)――――1
RS232Cケーブル――――1
ICEROM8/532説明書――――1

●コマンド一覧

G	実行
B	ブレークポインターセット
BC	〃　クリア
BL	〃　リスト
S	セット
F	メモリフィル
M	ムーブ
R	Sフォーマットリード
SR	シンボルファイルリード
SL	シンボル リスト
T	トレース
D	メモリーダンプ
L	ディスアッセンブル
X	レジスター表示

開発・製造

株式会社北斗電子
〒060 札幌市中央区南2条西6丁目 狸小路プラザハウス2F
TEL011-251-2736　FAX011-271-8468

Q

The C Users Journal Japan 8月21日 創刊

Cマーケットの成長、Cプログラマの成熟に欠かせない C言語プログラミング情報誌 The C Users Journal Japan 完全日本語翻訳版、創刊号予定目次

★Columns

Doctor C's Pointers *By Rex Jaeschke*
第1回 sizeofの微細側面を探る
Standard C *By P.J.Plauger*
第1回関数の宣言
How To Do It ····In C *By Robert Ward*
強力なデータカプセル化によってリングバッファを構築する
UNIXツール *By Kenji Hino*
第1回 リダイレクション・フィルター・パイプ
Questions & Answers *By Ken Pugh*

★Theme

Embedded SQL *By E. M. 'Bud' Pass, ph. D*
Cソース中へのSQLコマンドの組み方
Program Your DSP In C *By Ray Simar & Alan Davis*
DSPをCでプログラムする
UNIX/ MS-DOS Portability *By Vaugh Vernon*
UNIX/MS-DOS間のアプリケーション移植性の為の設計
UNIX/ VAX Differences *By John Ribar*
UNIXからVAXへの環境の移植

★Features

Writing Devices Drivers With Turbo C *By Robert Allen*
Turbo Cによるデバイスドライバの作成
Monitor Reveals Patterns *By S. Roggenkamp*
AMIGA モニタによって実行の様子を明らかにする
fscanf() As A Psuedo-Parser *By Thomas R. Clune*
fscanf()を用いた構文解析プログラム
Simple Text Output On All AppleIIgs *By T A. Deluca*
AppleIIgs上の簡単なテキスト出力
Programming GEM Displays *By Brian Pottorff*
GEMディスプレイのCによるプログラムの標準的方法
Type Ahead & Interrupts On UNIX *By S. Weinstein*
UNIXシグナルとターミナルコントロールパラメータ

★User Reports

Zortech C++ Comments *By Ron Burk & Helen Custer*
ゾーテックC++
Auto Flow-C Comments *By Bill Spees*
Cソースからフローチャートを自動生成するAuto Flow-C
Sherlock Comments *By Darryl Mataya*
Sherlock もうひとつのデバッガ
Amiga 'lint' Comments *By Glenn J. Wiorek*
Amigaユーザのための本物のlint
MINIX Comments *By Ken Graham*
UNIXのソースを＄89で買いたいと思いませんか

★C Users' Group Transactions

CROBOTS *By tom Poindexter*
戦術をプログラムしてプレーするゲーム'CROBOTS'
Micro PLOX *By Robert L.Patton*
ドット・マトリックス・グラフィックス用プロティング言語

★Product Review

C-Index+ Reviewed *By Neil Freeman*
B-Treeファイルの管理を提供するソフトウェアツール

●P. J. プローガー氏から読者へ●

The C Users Journalにようこそ。
本誌の日本語版を提供できることを嬉しく思います。日本は長らく、Cプログラミング言語の重要な市場でした。
現在、Cの市場は世界中で急速に成長しつつあります。
市場が激しく変化しているときには、タイムリーな情報が重要であることを私は理解しています。The C Users Journalを日本語で提供することによって、Cに関する最新のニュースを、あなたがたが、より早く入手できるようにと願っています。
本誌はCの幾つかの重要な側面に関する情報を提供します。まず、Cを使って日常のプログラミングの問題を解くための、新しい方法や実証済みの確実な方法を読者にご紹介します。
本誌には、パーソナル・コンピュータからROM化組み込みシステムに至るまでの、あらゆるジャンルのCのアプリケーションが登場します。その一部は、今日使われている内でも最も強力なシステムです。
また、CのANSIおよびISO規格について学べる連載記事があります。
各号で数名のCの権威者たちがコラムを執筆していますが、これらの人々はCの標準規格の草案の作成に深く関わった人たちです。標準Cに今後追加される拡張についても、誌上でもれなくご報告していきます。さらに、オブジェクト指向プログラミングや並行処理、ベクトル演算など、Cの方言における最もホットな話題も扱っていきます。
今日のCプログラミングには、見逃すことのできないエキサイティングな話題が沢山あります。読者各位が常に最先端にいることができるよう、本誌は最大の努力をするつもりです。
みなさまも、本誌に対するご要望などを、どしどしお寄せください。
日本の仲間からどんな声が寄せられるか、私は楽しみにしています。

定価1800円（本体1748円）

Publisher

VillageCenter Inc.

〒101 東京都千代田区神田神保町2－2－34
千代田三信ビル7F　株式会社ビレッジセンター
☎03－239－1950 FAX 03－221－1768
〒振替東京9－350878

Q

M

資料請求No.362

N

トータルな開発から部分開発まで!!

設計者の立場に立ち設計します。
喜ばれる設計、それが我社のモットーです。

◆各種CPUチップ(1chip CPUから32bit CPUまで)及び各種I/Oコントローラの回路設計。

◆I/O制御ソフトウェア、基本ソフト(MS-DOSデバイスドライバーなど)

※アセンブラ、C言語、FORTRAN、その他言語に対応します。

◆パターン設計は片面から多層板まで、CAD対応も可。(パターン設計のみの発注もお受けします。)

◆カスタムチップ化等のご相談にも応じます。

業務内容

◆ハードウェア開発　◆ソフトウェア開発

◆パターン設計　◆試作 etc…

見積りは、

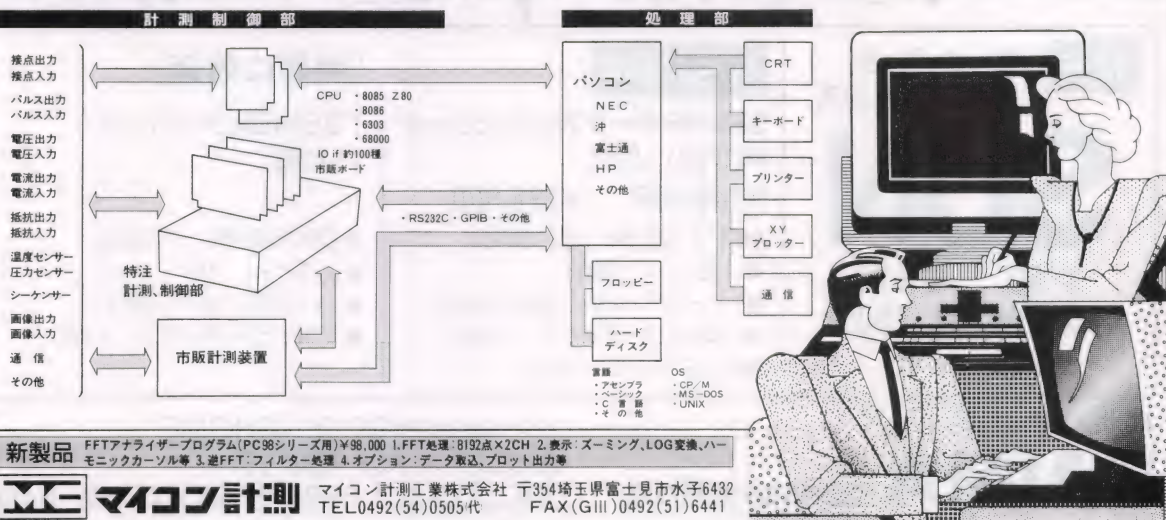
FAX.03-725-3886

TEL.03-725-1301代

リーダー商事株式会社

〒152 東京都目黒区緑が丘3-1-7 畑中ビル2F

資料請求No.363

パソコンによる計測・制御のシステム化を行なっております。

資料請求No.1363

外注先を、お探しの方へ。
納期厳守で高品質
仕様書
一枚の仕様書から
開発・設計・製作
少量多品種の生産も承ります。
まずは御一報頂きたいと存じます。
●ハードウェア　マイコン応用機器、計測器、試験器、医療機器周辺装置その他　●ソフトウェア　制御用ソフトウェア及び各種アプリケーションソフトウェア
異業種の方：構想だけでも御相談して頂ければ、御希望以上の製品を開発いたします。
(有)ホリウチ電子設計
〒213 川崎市宮前区有馬4－12－15
TEL.(044)853-2013　FAX.(044)853-2809

資料請求No.364

P

M N

マイクロソフトは、快適な環境と次代を切り拓く仕事を用意して、’90年代にジャンプしたいあなたをお待ちしています。

マイクロソフト株式会社は、多国籍企業「マイクロソフト」の日本法人。世界標準の製品を日本に届けるとともに、マイクロソフトの技術開発拠点として、次世代のソフトウェア開発にも積極的に取り組んでいます。

一人当り12m²以上のゆったりしたスペースと、全社員に1～3台の最先端のマシン環境。そして全世界のマイクロソフト社員の机と机を結ぶ電子メール。

マイクロソフト株式会社には、一人ひとりの興味を充分に満たし、持てる力を存分に発揮できる環境があります。今日の「世界標準」を受け継いで、明日の「世界標準」の創造にあなたの経験と実績を活かしてみませんか。あなたのチャレンジ精神を募集します。

募集要項

●募集職種

- OS/2・プレゼンテーションマネージャ・ウィンドウズの開発エンジニア
- OS/2テスティング エンジニア
- 言語製品・アプリケーション製品の開発エンジニア
- アプリケーション製品の製品計画エンジニア
- 製品マニュアル編集スタッフ
- 製品教育・セミナー企画/運営スタッフ及びマネージャ
- 言語製品及びアプリケーション製品のテクニカルサポートスタッフ
- 言語製品及びアプリケーション製品の品質管理スタッフ
- 広報宣伝・製品マーケティング・営業企画スタッフ
- ルートセールス営業スタッフ
- 法人セールス営業スタッフ
- システムソフトウェアのライセンス営業スタッフ
- 管理業務(人事、経理、総務)担当スタッフ

●能力・資格
特に問いません。30歳位までの方で希望職種の経験者か、興味・意欲ある方歓迎。英語力あれば尚可。男女は問いません。

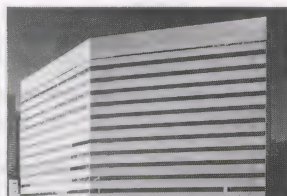
●給与
経験、年齢、能力、現行給与を充分考慮の上、当社規定により優遇します。中途入社によるハンディはいっさいありません。

●勤務地
原則として東京。職種により長期海外出張あり。

●待遇
昇給年1回(4月)。賞与年2回(6月、12月)。

●勤務時間
原則として9:00～17:30。フレックスタイム制あり。

●休日休暇
完全週休2日制、祝祭日、年次有給休暇(初年度12日)、夏期休暇、年末年始休暇、慶弔休暇など。

●福利厚生
通勤手当全額支給、各種社会保険完備、退職金制度、社員持株制度、財形貯蓄制度など。

1989年8月14日、
新社屋に移転しました。

会社概要

設立	1986年2月
資本金	1億円
売上高	46億円(1989年度実績)
代表取締役社長	古川 享
従業員数	120名(1989年7月現在)
事業内容	パソコン用ソフトウェア(OS、言語、アプリケーション)及びそれに関連する製品の開発・販売
事業所	東京都新宿区西新宿 米国本社/マイクロソフト・コーポレーション(ワシントン州レッドモンド)

応募方法

履歴書(写真貼付)及び職務経歴書を郵送してください。書類選考の上、面接の日程をお知らせします。
※応募の秘密は厳守します。
※応募書類は返却しません。
※入社日は相談に応じます。

連絡先
マイクロソフト株式会社
〒160 東京都新宿区西新宿7-5-25
Kビルディング
社長室（担当）:金井
電話 03-363-1200(代表)

〈技術者募集広告〉

R

〈技術者募集広告〉

TAMTEX

でてこい、ヒーロー。

襲いかかる巨大なドラゴンに立ち向かう彼は、わが社のヒーローのひとり。そしていま私たちは、彼を超える新しいヒーローの生みの親を求めています。21世紀に向かって、さらに創造的なアミューズメントツールに発展しようとしている、エレクトロニクスゲーム。
豊かな創造力とヒューマンな感性で、時代とともに変化する遊びの本質をとらえ、より個性的なレジャーライフの実現に情熱を注ぐことのできる技術者を待っています。タムテックスで、可能性に満ちたあなたの才能を存分に発揮して、ゲームソフト開発のヒーローを目指してください。

社員募集

■会社概要
設立/昭和61年3月
※株式会社ナナオの関連会社、アイレム株式会社の100%出資会社です
資本金/2,000万円
従業員数/14名(男11名、女3名)
平均年齢/24歳
代表者/代表取締役社長　高嶋由之
事業内容/ファミリーコンピュータ、PCエンジン等 コンピュータゲーム の企画・開発・制作

■募集要項
職種/ゲームソフトフログラマー
資格/20~26歳位迄 アッセンブラー、Z80、8086、6502等の経験者歓迎
給与/年齢、経験を考慮の上、当社規定により決定します。
例) 20歳㊊149,900円以上
待遇/昇給年一回、賞与年2回、交通費全額支給、各種社会保険完備
勤務時間/9:30~18:00
勤務地/広尾(南麻布)
休日/日・祝、隔週土曜休 年末年始、夏季、(年間102日)有給他各休暇あり。
応募/電話連絡の上、履歴書(写真貼付)持参でご来社下さい。 委細面談
※電話による問合せ歓迎
※応募の秘密厳守します。
交通/㊞日比谷線広尾駅下車歩12分
JR渋谷駅~バス便にて四ノ橋停1分

株式会社タムテックス
☎03-440-5881(代)
〒106 東京都港区南麻布3-19-23
オーク南麻布ビル14F

〈技術者募集広告〉

エンジニアの探求心や
好奇心を
絶えず刺激し、
無限の可能性を
提供する
アルテック・エンジニアリング。

まだスタートしたばかりの会社です。でも、確かなビジョンがあります。各種マシンの納入立ち合い・取付け、調整・保守をおこなう技術スタッフとアシスタントを募集しています。

バックボーンは2つの一流企業。

アルテック・エンジニアリングの母体は、一般産業機械（プラスチック及び紙加工成形機、コンピュータ関連周辺機器、真空蒸着機械等多数）を扱う専門商社アルテック（株）と、東証1部上場企業であるオーエム製作所の共同出資により設立された企業。

世界の最先端技術にタッチできます。

アルテックは、ドイツ、スイス、オランダ、イタリア等のヨーロッパ50数社、またアメリカ30数社と、世界各国メーカーを相手に取引している関係上、当社は、新機種のトレーニングのためエンジニアを海外に派遣する機会が多く、このことは、世界各国の最先端技術にタッチできるということであり、好奇心旺盛なエンジニアをきっと満足させるはずです。

広範囲な専門知識が身につきます。

フィルム関連押出機、ラミ・コーター、再生機及び帯電防止機器、食品加工機械、真空蒸着機械、コンピュータ関連機器……。
当社が扱う機械は、あらゆる機械に及んでいるので、幅広い機械を扱う変化のある仕事です。また、設計に興味があればそちらの分野へも進めます。当社はあなたの探究心を刺激します。

■株主構成
アルテック㈱55%、㈱オーエム製作所（東証一部上場）45%

■関連会社
アルテックリース㈱、大隈クラウスマファイ㈲

■主要取引先
花王、共同印刷、三洋電機、大日本印刷、東罐興業、東芝、凸版印刷、日本ＩＢＭ、ＴＤＫ、富士通、ＩＢＭ、ソニー、日立マクセル他大手企業多数

■関連会社アルテック㈱会社概要
設立／昭和51年5月　資本金／3000万円　年商／60億円（63年実績）従業員数／50名　事業内容／機械類及び電子情報計測機類の輸出入及び国内販売　事業所／東京本社、大阪営業所、ニューヨーク事務所（米国）、デュッセルドルフ事務所（西ドイツ）

◇

【事業内容】一般産業機械及び磁気メディア・コンピュータテープ周辺機器類等の保守管理ならびに産業機械の自動化ラインの設計・デザイン・ソフトウェア事業。
【職種】サービスエンジニア及びアシスタント
【資格】工業高校卒以上45歳位迄　※経験者優遇
【給与】18万円以上（住宅・食事手当含む）
例28歳）19万7000円（住宅・食事手当含む）
【待遇】昇給年1回、賞与年2回、交通費全額支給、各種社会保険完備、住宅・食事・役職手当、財形貯蓄、社内旅行あり
【休日休暇】完全週休2日制（土・日）、有給・慶弔・年末年始・夏季休暇
【勤務時間】9：00〜17：00
【勤務地】中央区日本橋本町
【応募】電話連絡の上、履歴書（写真貼付）持参
※応募の秘密厳守致します。
※入社日相談に応じます。
【採用関係連絡先】〒103 東京都中央区日本橋本町1－5－9 本町1丁目ビル6階 Tel 03-279-3691
【交通】銀座線三越前駅下車徒歩2分、総武線快速日本橋駅下車徒歩5分
【会社概要】設立●昭和62年9月　資本金●3000万円
年商●4000万円（63年実績）　従業員数／8名

☎03（279）3691担当／加畑まで。

アルテック・エンジニアリング株式会社　〒103 東京都中央区日本橋本町1－5－9本町1丁目共同ビル6階

<技術者募集広告>

あらゆる産業のハイテク化が進展するなか、エムテックの使命は各方面のニーズを的確にとらえ、ハード／ソフト両面にバランスのとれたシステムを提供していくことです。
マイクロコンピュータや画像処理分野で培ってきた豊富な経験。そして、後楽園エアドームの空気膜振動計測など、広範囲で活用されている「画像計測システムシリーズ」等を生み出した先進の技術力。
大手企業、官公庁、大学研究室からも熱い視線を集めるエムテックは、今後もオリジナル開発、そしてユーザーの希望に叶うシステム開発を通して未来に貢献していきます。

画像処理探険隊

事業内容　電子応用機器の製造・販売／①2次元・3次元移動体位置計測装置　②各種画像計測システム　③生産ライン計測・制御システム　④各種マイクロコンピュータ応用ハード・ソフト　⑤画像イメージプロセッサの開発及び応用に関する研究

■募集要項■

職種　①電子回路設計技術者
②ソフト開発技術者
③セールスエンジニア

資格　大学・専門学校卒以上　35歳迄

給与　経験・年齢を考慮の上優遇します。
例）89年度大卒実績／16万円
30歳／年収500万円(月額28万2500円、賞与4.5ヵ月含む、88年度実績)

待遇　昇給年1回、賞与年2回、交通費全額支給、職能・家族・残業・住宅手当有、各種社会保険完備、退職金制度・財形貯蓄制度有

勤務地　渋谷（技術開発室）五反田（本社）

勤務時間　9:00～17:30

交通　ＪＲ渋谷駅下車徒歩６分

休日休暇　完全週休２日制(土・日)、祝日、年末年始、夏季、有給休暇

応募　履歴書(写真貼付)を本社総務部小岩迄郵送下さい。追って面接日御連絡いたします。(電話連絡の上持参も可)
※入社日応相談。休日・夜間の面談可。どんな内容でもお問い合わせ歓迎します。ご希望の方には資料をお送りします。

来春新卒者同時募集中

株式会社エムテック

本　　社／東京都品川区東五反田1-25-13 神野商事ビル　Tel(03)449-3721(代)
技術開発室／東京都渋谷区渋谷3-27-10 第一久我屋ビル　Tel(03)498-7791(代)

Distribution Revolution

私たちはソフト、ハードにこだわらない。コンピュータを利用して、いかに流通業界を活発に、そしてスムーズに動かすか。そう、あくまで経営コンサルティングの一貫としてソフトを開発、必要とあればハードまで手掛ける。画期的なＰＯＳシステム、ネバーランドは、こうした企業理念から生れた一つの自社製品だ。今後も企業と企業、人と人とのつながりを大切に素晴らしい流通革命“Distribution Revolution”を起こしたいと思う。さらに成長したい人の成長する企業、ニッセイ・シーガル。

売上高昨年度比40％増

■会社概要■

設立／昭和60年11月　**資本金**／2500万円　**売上高**／４億円（63年度実績）
従業員数／20名　**関連会社**／㈱講談社経営総合研究所、日精電機㈱、㈱東京ベンチャーキャピタル
事業内容／POSシステムや、CAD／CAM用ソフトウェア等の開発・販売。自社ハードウェアの開発・販売
主要取引先／日本アイ・ビー・エム株式会社、キヤノン㈱、松下電器㈱、日本電気株式会社、他

■募集要項■

職種　①ＳＥ・プログラマー
（ＰＯＳ・通信システムのプログラミング）
②システムコンサルタント
（顧客へのコンピュータ導入指導）

資格　高卒以上、19～30歳位迄
※経験者優遇

給与　当社規定により優遇致します。
例）経験者　25歳
23万＋技能手当、職務手当

待遇　昇給年1回、賞与年2回、社会保険完備、社員持株制度有、住宅手当、食事補助、海外研修制度

休日休暇　完全週休２日・祝日、夏季、年末年始・有給、特別休暇有

勤務地　池袋

勤務時間　9:00～17:30

応募　電話連絡の上、履歴書持参。

交通　池袋駅北口徒歩３分。

株式会社ニッセイシーガル

〒171 東京都豊島区西池袋1-43-3 NISSEI BLD.　TEL.03-985-9760㈹ 担当/会田、窪山

R

協栄エレクトロニクス

帰ってきんしゃい！『よか博多』

次代のエレクトロニクスを担う
情熱人間を求めます。

■募集要項■
●職種 マイコン関連ソフト・ハードSE技術者/営業
●資格 高卒以上、35歳位迄の男女
●給与 経験、年齢を考慮の上、当社規定により優遇
●待遇 昇給年1回、賞与年2回、各種社会保険完備、退職金制度有、交通費全額支給
●勤務地 本社(福岡)
●休日・休暇 日曜、祝祭日、年末年始、夏季、有休、特別休暇
●応募 電話連絡の上、履歴書(写真貼付)を郵送下さい。追って面接日をご連絡致します。

■事業内容■
①FA関連装置の電子機器開発設計
②計測機器の開発設計
③通信系応用機器の開発設計
④マイコン機器販売
（ソフィアシステムズ、ワコム等）

Kyoei electronics
協栄興産㈱グループ
〒810 福岡市中央区渡辺通1丁目1番1号
電気ビル別館 ☎(092)761-6657
担当：吉村

アイデアを大きく育てませんか！

●Uターンで実力を発揮したい方
●ハード＋ソフトの両刀使いを自負される方
●BASICだけが恋人でない(C、ASMも出来る)方
●テストシステム開発に興味のある方
●DSP・画像・ISDN等経験者、又はやってみたい方

そして"新しいことにチャレンジする"同志を募集します。

■募集要項■
職種 ①電子回路技術者
②ソフトウェア技術者
資格 ●工高、高専、専門学校卒以上
●ソフト経験者(2～3年)
(情報処理1、2種又はマイコン技術者中級あれば尚可)
給与 経験・現在収入を考慮の上優遇
待遇 昇給年1回、賞与年2回
休日・休暇 日曜・祝日・第2、4土曜
夏期・年末年始休暇
年次有給休暇
応募 履歴書(写真貼付)を郵送下さい。追って面接日連絡致します。(電話連絡の上持参も可)

■会社概要■
●設 立：昭和58年5月
●資本金：2000万円
●事業内容：モジュールボード開発、PROMライタ、表示システム、テストシステム開発、ハード含ソフト開発及びサポート

株式会社ロジパック 〒438 静岡県磐田市国府台77-3 TEL(0538)32-2822㈹ FAX(0538)34-1082

お勧めの方の資料請求カードご記入法
姓名を必ず分けて、フリガナをふって下さい
郵便番号も必ずご記入下さい
1丁目1番1号は1-1-1と書いて下さい
部署名は、事業部は(事)と略し、部課係名まで詳しく書いて下さい
従業員数は全社の従業員数で1～8の中から選んでご記入下さい
CQ出版
インターフェース・広告
TEL 03・947・6311(代)
03・947・4949(直)
資料請求カード
IF
1989年4月号
R

医師の方の資料請求カードご記入法

姓名を必ず分けて、フリガナをふって下さい

郵便番号も必ずご記入下さい
1丁目1番1号は1-1-1と書いて下さい

部署名は、事業部は(事)と略し、部課係名まで詳しく書いて下さい

11 は10とご記入下さい
12 は99とご記入下さい
13 は11とご記入下さい
14 は勤務医の方は14、開業医の方は99とご記入下さい

CQ出版
インターフェース・広告
TEL 03・947・6311(代)
03・947・4949(直)

教職者の方の資料請求カードご記入法

姓名を必ず分けて、フリガナをふって下さい

郵便番号も必ずご記入下さい
1丁目1番1号は1-1-1と書いて下さい

学校名および学部名とともに役職名(教授、助教授、講師、助手)をもご記入下さい。

11 は8または9とご記入下さい
12 は99とご記入下さい
13 は14とご記入下さい
14 は14とご記入下さい
15 は無記入です

CQ出版
インターフェース・広告
TEL 03・947・6311(代)
03・947・4949(直)

仕事も遊びも真剣です。

「一太郎」「花子」を生み出した私たちジャストシステムの基本方針は"快適さの追求"。仕事も遊びも快適にするためには努力を惜しみません。没頭して働き、没頭して遊ぶ。仕事にも遊びにも集中できる環境がジャストシステムにはあります。新しい研究所も福岡・岡山に開設し(1989年春)、徳島、東京、大阪と活動範囲がますます拡がります。一番先に未来を見たいと思うバイタリティのある方をジャストシステムは求めています。

募集職種

職　種	職務内容	勤務地
プロジェクトマネージャー	ソフトウェア・ハードウェア開発プロジェクトのマネージメント	徳島・東京 大阪・岡山 福岡
営業マネージャー	セールスエンジニア／メーカーとの折衝、流通、販売店への技術指導	徳島・東京 大阪・福岡
ソフトウェア研究開発エンジニア	C言語・アセンブラ等高級言語の経験者 次世代OS、ウインドウシステム、コンパイラの開発 エキスパートシステムの開発 データベースの開発 ネットワークシステムの企画・設計・開発・管理 通信ソフト(各種プロトコル、LAN等)の開発設計 CAD技術者 情報システム設計 他、各種ソフト開発	徳島・東京 大阪・岡山 福岡
ハードウェア研究開発エンジニア	IC・LSI設計(デジタル・アナログ) ASIC設計 半導体プロセス開発 設計支援ツール開発 半導体装置技術、メンテナンス 半導体検査技術、品質保証 CPU周辺コントローラ等ハードウェア回路設計 他、コンピュータ・ディスプレイ周辺機器の開発	徳島・東京 大阪・岡山 福岡
企画マネージャー	ソフトウェア・ハードウェアの製品企画及び製品企画プロジェクトのマネージメント	徳島・東京 大阪
広報マネージャー	パブリシティの制作(原稿、ビデオ、ポップ等)／製品外装設計、制作ディレクター	徳島・東京

*以上、経験者を募集中ですが、未経験者も可。

■会社概要
創　　立／1979年7月
資 本 金／1千万円
売 上 高／36億7千万円(1988年4月期実績)
70億円(1989年4月期見込み)
代表取締役社長／浮川和宣
社 員 数／330名
平均年齢／26歳
事業内容／パソコン・ワークステーション用ソフトウェア・ハードウェアの研究開発及び販売
事 業 所／徳島本社、ジャストシステム研究所〔東京、大阪、岡山、福岡、USAコロンビア〕、営業所〔東京、大阪、福岡〕、サポートセンター〔東京、大阪〕、新居浜事業所
(岡山、福岡は今春開設予定)

■募集要項
職　　種／別表参照
給　　与／年齢・経験・能力等を考慮の上、当社規定により優遇いたします。
入社後は、年齢・実績にしたがい昇給いたします。
1988年4月大卒初任給17万円。
年俸例：30歳プロジェクトマネージャー700万円
勤 務 地／別表参照
待　　遇／昇給年1回、賞与年2回
福利厚生／交通費全額支給、各種社会保険完備、家族手当他諸手当あり、借り上げ社宅・独身寮(県外者)、退職金制度
休日休暇／完全週休二日制、祝日、有給休暇、慶弔、夏季・年末年始(各10日、1988年実績)
勤務時間／9：00～17：30
※出向・派遣業務は一切ありません。

■応募方法
◎履歴書(写真貼付)と職務経歴書(書式自由、希望職種を朱記して下さい)をご郵送下さい。書類選考の上、面接・試験日をお知らせします。◎会社資料をご希望の方は、ご連絡下さい。◎仕事の関係ですぐに応募できない方や遠方にお住まいの方でも、一人一人のご希望や条件を考慮し、ベストの状態で転職できるようにご相談に応じます。◎応募の秘密は厳守いたします。

■採用関係連絡先・書類送付先
〒770 徳島市沖浜東3丁目46番地
TEL.(0886)22-7240(直通)
担当／岩田、小林
株式会社ジャストシステム

R

インターフェース 九月号
昭和五四年五月二日第三種郵便物認可 平成元年九月一日発行（毎月一回一日発行） 第十五巻 第九号 通巻一四八号
定価六二〇円（本体六〇二円）
CQ出版社 〒170 東京都豊島区巣鴨1-14-2 ☎(03)947-6311㈹ 雑誌 01619-9

インターフェース Interface
'89-9
No.148
●特集 続OS-9/68K 徹底解剖(上級編)
CQ出版社